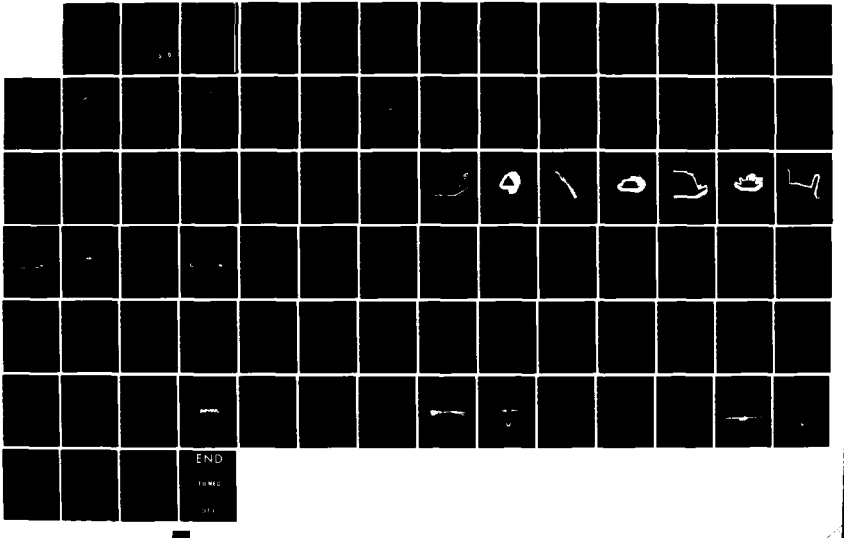
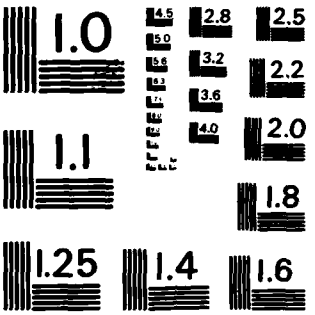
AD-A148 059 GENERATION OF THREE DIMENSIONAL BODY FITTED COORDINATES USING HYPERBOLIC.. (U) STANFORD UNIV CA DEPT OF AERONAUTICS AND ASTRONAUTICS J L STEGER MAR 84 1/1

UNCLASSIFIED AFOSR-TR-84-1022 AFOSR-82-0254 F/G 20/4 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

(2)

1a. REPORT SECURITY CLASSIFICATION

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION

AD-A148 059

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)
AFOSR-TR- 84-1022

6a. NAME OF PERFORMING ORGANIZATION
Department of Aeronautics & Astronautics
Stanford University
Stanford, California 94305

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION
AFOSR/NA

6c. ADDRESS (City, State, and ZIP Code)

7b. ADDRESS (City, State, and ZIP Code)
Building 410
Bolling AFB, DC 20332-6448

8a. NAME OF FUNDING/SPONSORING ORGANIZATION
AFOSR

8b. OFFICE SYMBOL (If applicable)
NA

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
AFOSR-82-0254

8c. ADDRESS (City, State, and ZIP Code)
Building 410
Bolling AFB, DC 20332

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61102F			

11. TITLE (Include Security Classification)
Generation of Three Dimensional Body Fitted Coordinates Using Hyperbolic Partial Differential Equations

12. PERSONAL AUTHOR(S)
Joseph L. Steger

13a. TYPE OF REPORT	13b. TIME COVERED FROM ____ TO ____	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
FINAL		March 1984	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

See back

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
DR JAMES WILSON	(202) 767-4935	NA

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

The purpose of this research has been to further develop a simple efficient grid generation procedure for external aerodynamics aplications. The grid generation scheme is based on solving hyperbolic partial differential equation constraints of grid angularity and mesh incremental volumes. The grid generation scheme has been previously used in two dimensional applications to generate grids about smooth body shapes. The main thrust of this AFOSR supported research has been to extend the hyperbolic partial differential equation procedure to three dimensional applications and to study ways of applying the procedure to body shapes that have discontinuous derivatives.

The main part of this report, Part I, is devoted to describing the three dimensional hyperbolic grid generator. This Section first reviews the hyperbolic grid generation procedure in two dimensions and then describes the extension to three dimensions. The numerical solution algorithm, mesh control functions and treatment of axis singularities are then described. Computed meshes and even are calculated flow fied result are shown to help evaluate the grids. Part I of this report is essentially a self contained technical report that is being readied for submission to a Journal.

Part II of this report is both brief and sketchy in its presentation. In this section we describe some of our success in treating bodies with sharp edges and bodies that are exceptionally concave. The hyperbolic grid generation tends to propogate discontinuities and some bodies are not compatible with the imposed orthogonality and volume constraints that are user imposed. When this happens, the hyperbolic grid generator breaks down. What must be done in this case is to relax these constraints. We have had some success doing this, but this process remains still an art form.

The last part of this report describes a flow field algorithm development. During the course of this research we had some considerable interaction with AFWAL, and at one point became 'side-tracked' into a successful approach of improving the efficiency of our general implicit Euler and Navier-Stokes code. Part III of this paper contains a preliminary paper describing this development.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist | Avail and/or Special	
A-1	

GENERATION OF THREE DIMENSIONAL BODY FITTED COORDINATES USING HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS

Joseph L. Steger

Department of Aeronautics and Astronautics

Stanford University

Stanford, California 94305

March 1984
Final Report for Grant AFOSR-82-0254

FOREWORD

The purpose of this research has been to further develop a simple efficient grid generation procedure for external aerodynamics aplications. The grid generation scheme is based on solving hyperbolic partial differential equation constraints of grid angularity and mesh incremental volumes. The grid generation scheme has been previously used in two dimensional applications to generate grids about smooth body shapes. The main thrust of this AFOSR supported research has been to extend the hyperbolic partial differential equation procedure to three dimensional applications and to study ways of applying the procedure to body shapes that have discontinuous derivatives.

The main part of this report, Part I, is devoted to describing the three dimensional hyperbolic grid generator. This Section first reviews the hyperbolic grid generation procedure in two dimensions and then describes the extension to three dimensions. The numerical solution algorithm, mesh control functions and treatment of axis singularities are then described. Computed meshes and even are calculated flow fied result are shown to help evaluate the grids. Part I of this report is essentially a self contained technical report that is being readied for submission to a Journal.

Part II of this report is both brief and sketchy in its presentation. In this section we describe some of our success in treating bodies with sharp edges and bodies that are exceptionally concave. The hyperbolic grid generation tends to propogate discontinuities and some bodies are not compatible with the imposed orthogonality and volume constraints that are user imposed. When this happens, the hyperbolic grid generator breaks down. What must be done in this case is to relax these constraints. We have had some success doing this, but this process remains still an art form.

The last part of this report describes a flow field algorithm development. During the course of this research we had some considerable interaction with AFWAL, and at one point became 'side-tracked' into a successful approach of improving the efficiency of our general implicit Euler and Navier-Stokes code. Part III of this paper contains a preliminary paper describing this development.

PART I. GENERATION OF THREE DIMENSIONAL BODY FITTED COORDINATES USING HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS

INTRODUCTION

Body conforming curvilinear grids are often used in finite difference flow field simulations. One reason for this is that the application of boundary conditions can be simplified in finite difference calculations because grid lines coincide with boundary lines. This is especially important in high Reynolds number viscous flow simulation in which high flow gradients near the body surface must be resolved.

The task of generating a satisfactory body conforming coordinate system is not easy. The grids must not be too distorted, they should have smooth variation, and they should be clustered to flow field action regions – typically near boundary surfaces. Moreover, the grids should be generated in an automatic manner that requires a minimum of user input.

One approach for generating body conforming grids with minimum user input has been to solve a set of partial differential equations. In this technique level lines of $\xi(x,y,z)$, $\eta(x,y,z)$, and $\varsigma(x,y,z)$ that have monotone variation are sought as a solution of a set of partial differential equations. Generally values of ξ, η and ς are user specified on the boundary surface and constraints expressed as differential equations are used to develop the grid away from the boundaries. The most popular such approach requires the solution of a set of elliptic equations that satisfy the maximum principle [1-5], however, hyperbolic [6,7] and parabolic [8] governing equations have been used as well, at least in two dimensional applications.

In this report one way of extending the hyperbolic grid generation method of Steger and Chaussee [6] to three dimensions is developed. In two dimensions the two differential constraints

$$\xi_x \eta_x + \xi_y \eta_y = 0 \qquad ((1a))$$

$$\xi_x \eta_y - \xi_y \eta_x = (\Delta V)^{-1} \qquad ((1b))$$

or in ξ, η computational space

$$x_\xi x_\eta + y_\xi y_\eta = 0 \qquad ((2a))$$

$$x_\xi y_\eta - x_\eta y_\xi = \Delta V \qquad ((2b))$$

have been solved by marching in η from an initial data plane $\eta(x,y) = \text{constant}$. The first equation is a constraint of orthogonality. The second equation controls the mesh spacing with the user specifying the mesh control volume ΔV (actually area in two dimensions). A linearized version of equations (2) is readily shown to be

hyperbolic and suitable for marching in η. Equations (2) are solved in computational space to give the x, y location of the ξ = constant and η = constant grid lines.

The two partial differential equations, expressed as either Equations (1) or Equations (2), have been referred to as a mesh cell volume procedure for grid generation. In the next section a three dimensional extension of this procedure is developed.

THREE DIMENSIONAL GRID GENERATION EQUATIONS

A body fitted exterior grid about an arbitrary closed boundary surface is desired. Only a simple topology such as that illustrated in Figure (1) will be considered here. The body surface is chosen to coincide with $\varsigma(x, y, z) = 0$ and the surface grid line distributions of ξ = constant and η = constant are user specified. The outer boundary $\varsigma(x, y, z) = \varsigma_{max}$ is not specified, it is only required to be sufficiently far removed from the inner boundary. Using ς as the marching direction, partial differential equations are sought which produce planes of constant ξ, η and ς to form a nonsingular mesh system.

An extension of the mesh cell volume procedure to three dimensions is proposed. In three dimensions, however, there are three orthogonality relations and one cell volume constraint. At any point four equations are available to predict the three unknowns x, y and z so one equation must be discarded. Because ς is the marching direction it is natural to use only the two orthogonality relations that involve ς, this leads to the governing equations

$$x_\xi x_\varsigma + y_\xi y_\varsigma + z_\xi z_\varsigma = 0 \qquad ((3a))$$

$$x_\eta x_\varsigma + y_\eta y_\varsigma + z_\eta z_\varsigma = 0 \qquad ((3b))$$

$$x_\xi y_\eta z_\varsigma + x_\varsigma y_\xi z_\eta + x_\eta y_\varsigma z_\xi - x_\xi y_\varsigma z_\eta - x_\eta y_\xi z_\varsigma - x_\varsigma y_\eta z_\xi = \Delta V \qquad ((3c))$$

or with $\vec{r}$ defined as $(x, y, z)^t$

$$\vec{r}_\xi \cdot \vec{r}_\varsigma = 0, \quad \vec{r}_\eta \cdot \vec{r}_\varsigma = 0, \quad \left|\frac{\partial(x, y, z)}{\partial(\xi, \eta, \varsigma)}\right| = J^{-1} = \Delta V$$

The first two equations represent orthogonality relations between ξ and ς and between η and ς, while the last equation is the volume or finite Jacobian constraint.

Equations (3) comprise a system of nonlinear partial differential equations in which x, y and z are specified as initial data at $\varsigma = 0$. As developed below, linearization and analysis of Equations (3) about a nearby known state reveals that the system is hyperbolic with ς as the marching direction.

Let x^0, y^0, z^0 represent a nearby known state so that

$$x = x^0 + (x - x^0) = x^0 + \tilde{x}$$
$$y = y^0 + \tilde{y}$$
$$z = z^0 + \tilde{z}$$

where $\tilde{x}, \tilde{y}$ and $\tilde{z}$ are small. Substitution of these expressions into Equations (3) and elimination of products of tilde terms results in the locally linearized system

$$A_0(\vec{r}-\vec{r}_0)_\xi + B_0(\vec{r}-\vec{r}_0)_\eta + C_0(\vec{r}-\vec{r}_0)_\zeta = \vec{f} \qquad ((5))$$

with

$$A = \begin{pmatrix} x_\zeta & y_\zeta & z_\zeta \\ 0 & 0 & 0 \\ (y_\eta z_\zeta - y_\zeta z_\eta) & (x_\zeta z_\eta - x_\eta z_\zeta) & (x_\eta y_\zeta - x_\zeta y_\eta) \end{pmatrix} \qquad ((6a))$$

$$B = \begin{pmatrix} 0 & 0 & 0 \\ x_\zeta & y_\zeta & z_\zeta \\ (y_\zeta z_\xi - y_\xi z_\zeta) & (x_\xi z_\zeta - x_\zeta z_\xi) & (x_\zeta y_\xi - x_\xi y_\zeta) \end{pmatrix} \qquad ((6b))$$

$$C = \begin{pmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ (y_\xi z_\eta - y_\eta z_\xi) & (x_\eta z_\xi - x_\xi z_\eta) & (x_\xi y_\eta - x_\eta y_\xi) \end{pmatrix} \qquad ((6c))$$

$$\vec{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad \vec{f} = \begin{pmatrix} -\frac{\partial \vec{r}}{\partial \xi} \cdot \frac{\partial \vec{r}}{\partial \zeta} \\ -\frac{\partial \vec{r}}{\partial \eta} \cdot \frac{\partial \vec{r}}{\partial \zeta} \\ \Delta V - \Delta V_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \Delta V - \Delta V_0 \end{pmatrix} \qquad ((6d))$$

Let $\vec{R} \equiv \vec{r} - \vec{r}_0$, then (5) is rewritten as

$$A_0\vec{R}_\xi + B_0\vec{R}_\eta + C_0\vec{R}_\zeta = \vec{f} \qquad ((7))$$

Now C_0^{-1} exists unless $(\Delta V)^{-1} \to \infty$, which we will not impose, so (7) can be rewritten as

$$C_0^{-1}A_0\vec{R}_\xi + C_0^{-1}B_0\vec{R}_\eta + \vec{R}_\zeta = C_0^{-1}\vec{f} \qquad ((8))$$

Although the verification is nontrivial, $C_0^{-1}A_0$ and $C_0^{-1}B_0$ are found to be symmetric matrices (this was carried out by Dennis Jesperson of the NASA Ames Research Center, who used MACSYMA). The linearized system Equation (8) is therefore hyperbolic and can be marched with ζ serving as the "time–like" direction.

It can be pointed out that an analysis was attempted for the three orthogonality relations alone. These equations, however, are found to be improperly posed for marching with initial data in ζ. Indeed, as best as we can discern, the three relations do not lend themselves to unique solutions regardless of the type of boundary conditions specified.

SOLUTION PROCEDURE

The nonlinear system of grid generation equations given by Equations (3) are solved with a noniterative implicit finite difference scheme. An unconditionally stable implicit scheme is chosen so the marching step size in ζ can be arbitrarily selected based only on considerations of accurately generating the grid. Iterative solution of the nonlinear grid generation equations is avoided by expanding the equations about the previous marching step. As a consequence Equation (7) is solved with the nearby known state 0 taken from the previous ζ step.

a) Numerical Method

Let $\Delta\xi = \Delta\eta = \Delta\zeta = 1$ such that $\xi = j-1, \eta = k-1$ and $\zeta = l-1$. Central spatial differencing of Equations (5) in ξ and η with first order backward implicit differencing in ζ leads to

$$A_l\delta_\xi(\vec{r}_{l+1} - \vec{r}_l) + B_l\delta_\eta(\vec{r}_{l+1} - \vec{r}_l) + C_l\nabla_\zeta\vec{r}_{l+1} = \vec{g}_{l+1} \qquad ((9))$$

where

$$\vec{g}_{l+1} = \begin{pmatrix} 0 \\ 0 \\ \Delta V_{l+1} \end{pmatrix}$$

and

$$\delta_\xi\vec{r}_j = \frac{\vec{r}_{j+1} - \vec{r}_{j-1}}{2}, \quad \delta_\eta\vec{r}_k = \frac{\vec{r}_{k+1} - \vec{r}_{k-1}}{2}$$

$$\nabla_\zeta\vec{r}_{l+1} = \vec{r}_{l+1} - \vec{r}_l$$

Note that $C_l\delta_\zeta\vec{r}_l$ was subtracted from $\vec{f}_{l+1}$ to produce $\vec{g}_{l+1}$ in the above. Throughout only those indices that change are indicated, thus $r_{l+1} \Rightarrow r_{j,k,l+1}, r_{j+1} \Rightarrow r_{j+1,k,l}$ etc.

Multiplying through by C_l^{-1} gives

$$C_l^{-1}A_l\delta_\xi(\vec{r}_{l+1} - \vec{r}_l) + C_l^{-1}B_l\delta_\eta(\vec{r}_{l+1} - \vec{r}_l) + I(\vec{r}_{l+1} - \vec{r}_l) = C_l^{-1}\vec{g}_{l+1} \qquad ((10))$$

where I is the identifying matrix. To reduce the inversion cost the difference equations are approximately factored as

$$(I + C_l^{-1}A_l\delta_\xi)(I + C_l^{-1}B_l\delta_\eta)(\vec{r}_{l+1} - \vec{r}_l) = C_l^{-1}\vec{g}_{l+1} \qquad ((11))$$

so that $\vec{r}_{l+1}$ is obtained by solving sequences of one–dimensional–like block tridiagonal systems

$$(I + C_l^{-1}A_l\delta_\xi)\vec{g}^*_{l+1} = \vec{g}_{l+1} \qquad ((12a))$$

$$(I + C_l^{-1}B_l\delta_\eta)\nabla_\zeta\vec{r}_{l+1} = \vec{g}^*_{l+1} \qquad ((12b))$$

$$\vec{r}_{l+1} = \vec{r}_l + \nabla_\zeta \vec{r}_{l+1} \qquad ((12c))$$

In practise numerical dissipation terms are added in the ξ and η directions. Typically we have used a combination of fourth and second differences which are explicitly and implicitly included into the basic algorithm as

$$[I + C_l^{-1} A_l \delta_\xi - \epsilon_i (\Delta\nabla)_\xi][I + C_l^{-1} B_l \delta_\eta - \epsilon_i (\Delta\nabla)_\eta](\vec{r}_{l+1} - \vec{r}_l) = C_l^{-1} \vec{g}_{l+1} - [\epsilon_e (\Delta\nabla)_\xi^2 + \epsilon_e (\Delta\nabla)_\eta^2]\vec{r}_l$$

where ϵ_e is order $\frac{1}{16}$ and ϵ_i is 2.5 or more larger than ϵ_e. As an alternative to fourth differences, second order terms such as

$$\epsilon |\Delta\nabla \vec{r}_l| \Delta\nabla \vec{r}_l$$

have been used both implicitly and explicitly.

The coefficient matrices A_l, B_l and C_l contain ξ and η derivatives which are formed using central differences. These matrices also contain derivatives for x_ζ, y_ζ and z_ζ which are obtained from Equations (3) in terms of ξ and η derivatives. That is, Equations (3) are linear in the unknowns x_ζ, y_ζ and z_ζ. They are easily solved for as

$$\begin{pmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{pmatrix} = \frac{\Delta V}{(DetC)} \begin{pmatrix} x_\xi z_\eta - y_\eta z_\xi \\ x_\eta z_\xi - x_\xi z_\eta \\ x_\xi y_\eta - x_\eta y_\xi \end{pmatrix} \qquad ((13))$$

with

$$Det(C) = (y_\xi z_\eta - y_\eta z_\xi)^2 + (x_\eta z_\xi - x_\xi z_\eta)^2 + (x_\xi y_\eta - x_\eta y_\xi)^2$$

Note that $\Delta V / \sqrt{(x_\zeta^2 + y_\zeta^2 + z_\zeta^2)} = \mathrm{Det}(C)$ so that $\mathrm{Det}(C)$ will be zero if and only if the user specified $\Delta V = 0$. Hence, C^{-1} will exist.

b) Cell Volume Specification

The user has control of the grid by means of the initial surface point distribution and by specification of the cell volumes, $\Delta V_{j,k,l}$. Through the cell volumes the extent and clustering of the grid can be essentially controlled. Because the cell volume at each point must be specified, it is clear that the user must devise some kind of method for determining volumes. There are many possibilities, here one such approach is illustrated.

Suppose we had a sphere to grid. A reasonable grid might have uniform angle spacing and have a radial grid distribution that is exponential. For this special geometry and grid we can analytically determine the control volumes by a simple formula. Now take the problem at hand, perhaps an aircraft fuselage, which we want to mesh as a warped spherical–like grid. We can find a sphere that has the same surface area as our fuselage and use the grid cell volumes of the sphere to

specify the cell volumes of the fuselage grid. However, the fuselage will not have the same kind of surface area distribution as a sphere with equal angle distribution. So here we need an adjustment, something like

$$\Delta V_{j,k,l} = (\Delta V_{j,k,l})_{sphere} \quad \left[(1-\delta) + \frac{(\Delta A_{j,k})_{sphere}}{(\Delta A_{j,k})_{fuselage}}\,\delta\right] \qquad ((14))$$

where $\delta \rightarrow 1$ for small l and $\delta \rightarrow 0$ for large l. That is, the volumes would be adjusted initially to the local boundary surface increments. But as we march out the uniform spherical volumes would gradually be specified. Such an approach has been used, and, as a result, the far field portion of the grid tends to be uniformly spherical. The results shown later will illustrate this behavior.

C) Axis Treatment

A coordinate singularity will be encountered whenever a regular grid is mapped over a closed body such that $\zeta = 0$ corresponds to the body surface. Here we will generate warped spherical grids, so equation (5) becomes singular at the axis. Using ζ as the marching direction with ξ and η as sketched in Fig.2 (i.e. ξ from pole to pole and η equatorial) then the axis at $\xi = 0$ and $\xi = \xi_{max}$ represent singularities. In particular, η derivatives approach zero at the pole and equations (2b) and (2c) are lost. If there are KMAX points in the η-direction, however, the difference equation corresponding to Eq.(2a) can be imposed KMAX times to solve for three axis unknowns, namely x,y, and z. Thus the axis points are overdetermined and can be solved for via a generalized inverse scheme. Such an approach has been coded. However, it is difficult to implement implicitly, and it has not proved to be reliable for severely distorded axis cases, for example, when the axis extends from a wing tip.

Rather than try to compute the axis as the solution evolves, we have instead prespecified the axis position. For now a fixed straight axis has been used that is aligned with a coordinate. Typically a y-coordinate line has been chosen so that x and z are zero along the axis. Values of y along the axis are obtained as part of the solution process by imposing the orthogonality conditon, which in this degenerate case is simply $y_\eta = 0$. This condition is imposed using second order accurate three point differencing, and it is implicitly implemented into the η-block tridiagonal solution process using a slightly modified block solver.

RESULTS

To demonstrate the hyperbolic grid solver a series of grids were generated about ellipsoidal and 'super ellipsoidal' body shapes. The body generating function is given by

planform

$$(x/x_{max})^n + (y/y_{max})^n = 1$$

thickness

$$(y/y_{max})^m + (z/z_{max})^m = 1$$

profile

$$(x/x_{max})^l + (z/z_{max})^l = 1$$

where n, m and l are 2 for ellipsoids and 4 or 6 or 8 etc. for super ellipsoids. By chosing very high aspect ratios the ellipsoid can mimick a practical wing planform, or, for more moderate aspect ratios, a fuselage. In either case an axis singularity exists.

The right half of an elliptic wing planform with the user specified body surface grid point distribution is shown in Fig.2. This 'wing' has a 16:1 planform aspect ratio with a 20 per cent thick ellipse serving as the airfoil. Thus the body is an ellipsoid with ratios 16 : 1 : 1/5. A uniform grid spacing of 1/2 per cent thick maximum chord was specified as the first grid spacing off the body. Figures.4 to 6 show various segments of the generated grid for ξ and η = constant planes. Figure 4 shows an η =constant plane (here in the plane of the wing trailing edge) at the wing tip. As seen, the grid is very smooth, uniform off the body, and the axis at $x = 0$ is well behaved. A lower half of the grid at the wing midspan ($y = 0$) is shown in Fig. 5. This is a ξ = constant plane. Again the smoothness and grid clustering control is illustrated. Also in the far field the grid is tending to be spherical because spherical mesh incremental volumes have been specified. Finally, views above the wing in the $x = 0$ plane are shown in Figs. 6a and 6b. These views are focused at the wing tip and again illustrate satisfactory axis treatment.

A similar grid was computed as illustrated by Figs. 7 to 10. The wing in this case used a more realistic airfoil shape that has a thickness ratio of 15 per cent. The grid views Figs. 7 and 9 give an indication of the wing and airfoil section. Note that the airfoil has a rounded trailing edge.

It must be remarked that if the airfoil trailing edge radius is continuously reduced that the method breaks down. "Pretty bows" get tied within the grid. Provided the trailing edge radius is not zero, break down can usually be avoided by clustering grid points in this region and adjusting the specified volumes. In lieu of good surface clustering functions one can sometimes obtain an adequately resolved trailing edge by generating an overall much finer grid than what is desired for the flow solver. In this case we simply discard, say, every other grid point to obtain the final grid. Because the hyperbolic grid solver is quite efficient, we can readily operate in this manner. Ultimately the scheme does break down when subject to a truely sharp trailing edge.

The above grids have grid spacing that is adequate for inviscid flow simulations. The hyperbolic grid generation procedure can also directly generate grids suitable for high Reynolds number viscous flow simulation by simply having the user specify a much finer mesh spacing going out away from the surface. In Figs. 11 and 14 we

show views of a grid generated about a bluff body fuslage. The body in this case is defined as a 2 : 1 ellipsoid in the x-y and y-z planes (see Fig. 11). In the x-z plane a 1 : 1 super ellipsoidal cross section in specified by seting $m = 4$ (see Fig. 14). The first grid spacing off the body in this case was specified as 0.00002 of the body cross sectional diameter. Magnified grid lines coming into the axis are shown in Fig. 13 to indicate this fine grid spacing just off the body.

Although a less interesting configuration, a similar viscous grid was generated about a 6 : 1 : 1 ellipsoid. To demonstrate that the grids being generated with this approach are indeed useful, Figs. 15 to 16 show calculated particle paths and surface 'oil flow' simulations on this 6 : 1 : 1 ellipsoid. These Naiver Stokes calculations were carried out by T.H. Pulliam using ARC3D. The free stream conditions were chosen as Mach number of 0.74, angle of attack of 25 deg., and 0 yaw angle with a plane of symmetry imposed. The Reynolds number was 44×10^6 based on the diameter and laminar flow was assumed.

CONCLUSION

A procedure has been developed for generating body fitted coordinates in three dimensions using hyperbolic partial differential equations. In this report the scheme has been used to generate warped spherical-like grids about simple ellipsoidal wing and fuslage configurations. The hyperbolic grid generator can fail whenever the body surface is discontinuous or the user specified surface grid distribution is too irregular. For simple continous body shapes, however, the hyperbolic grid generation scheme can be very fast and reliable. It requires a minimum of user interaction, and it can be used to generate grids suitable for either inviscid or viscous flow simulation.

REFERENCES

1) Chu, W. H. Development of a general finite difference approximation for a general domain. J. Comp. Phys. Vol. 8, (1971), 392-408.

2) Godunov, S. K. and G. P. Prokopov, The use of moving meshes in gas-dynamical computations, USSR Comput. Math. Phys., 12, Vol. 2 (1972), 182-195.

3) Thompson, J. F., F. C. Thames, and C. M. Mastin, Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies, J. Comp. Phys., Vol. 15, (1974), 299-319.

4) Thompson, J. F., Elliptic grid generation, in Numerical Grid Generation, Joe F. Thompson, ed. North-Holland, New York, (1982).

5) Sorenson, R. L. and J. L. Steger, Grid generation in three dimensions by Poisson equations with control of cell size and skewness at boundary surfaces. in Advances in Grid Generation, ASME FED-Vol. 5,(1983) 181-187.

6) Steger, J. L. and D. S. Chaussee, Generation of body-fitted coordinates using hyperbolic partial differential equations, SIAM J. Sci. Stat. Comput., Vol. 1, (1980), 431-437.

7) Starius, G. Constructing orthogonal curvilinear meshes by solving initial value problems, Numerische Mathematik, Vol. 28, (1977), 25-48.

8) Nakamura, S. Marching grid generation using parabolic partial differential equations, in Numerical Grid Generation, Joe F. Thompson, ed. North-Holland, New York, (1982).

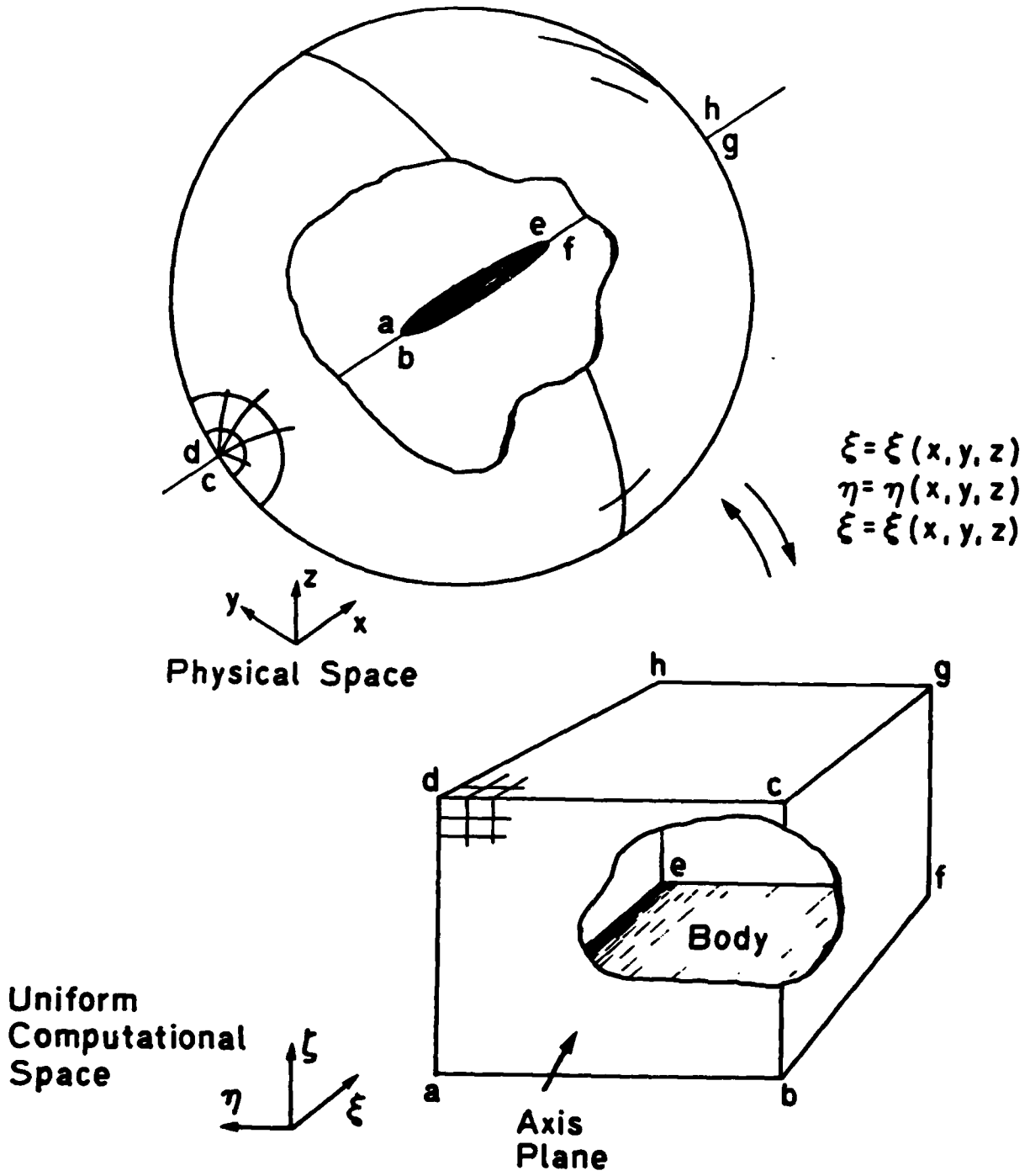


Figure 1. Well-Ordered Warped Spherical Grid Mapping

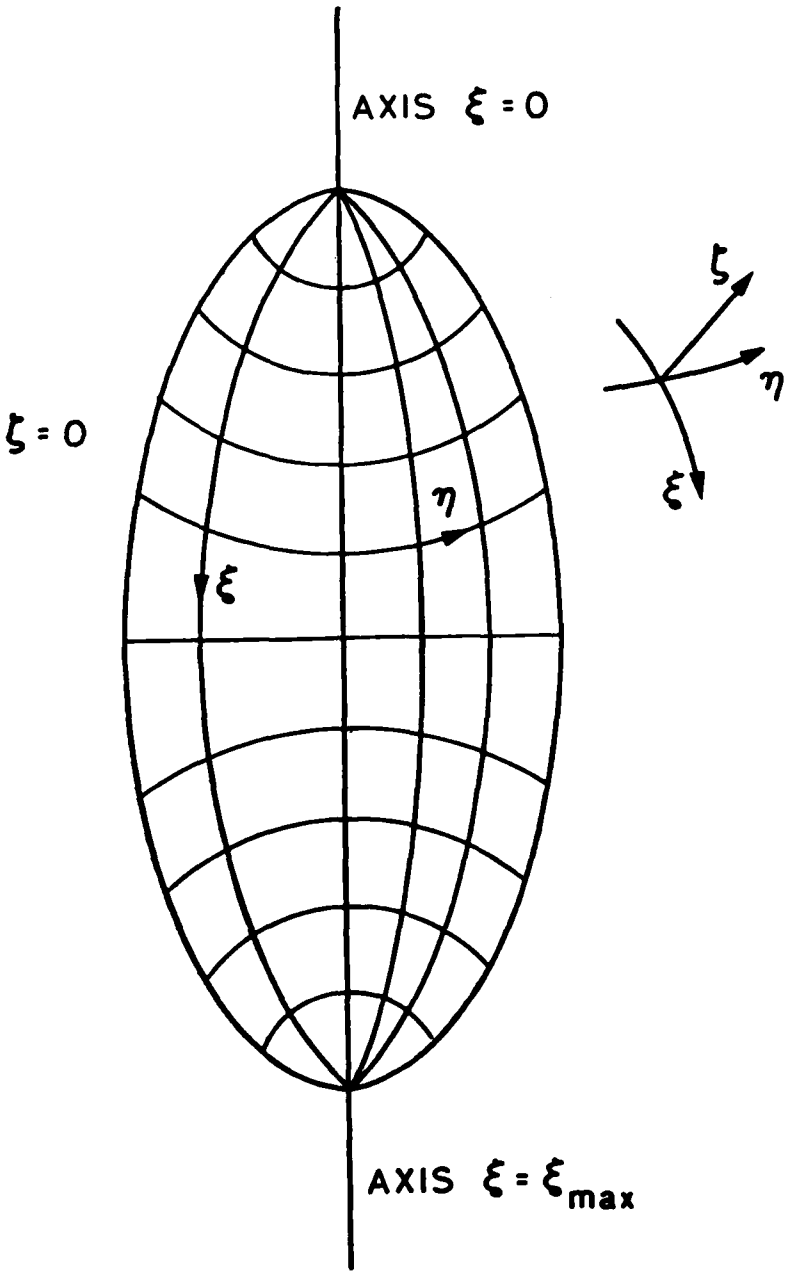


Figure 2. Surface Distribution at $\varsigma = 0$

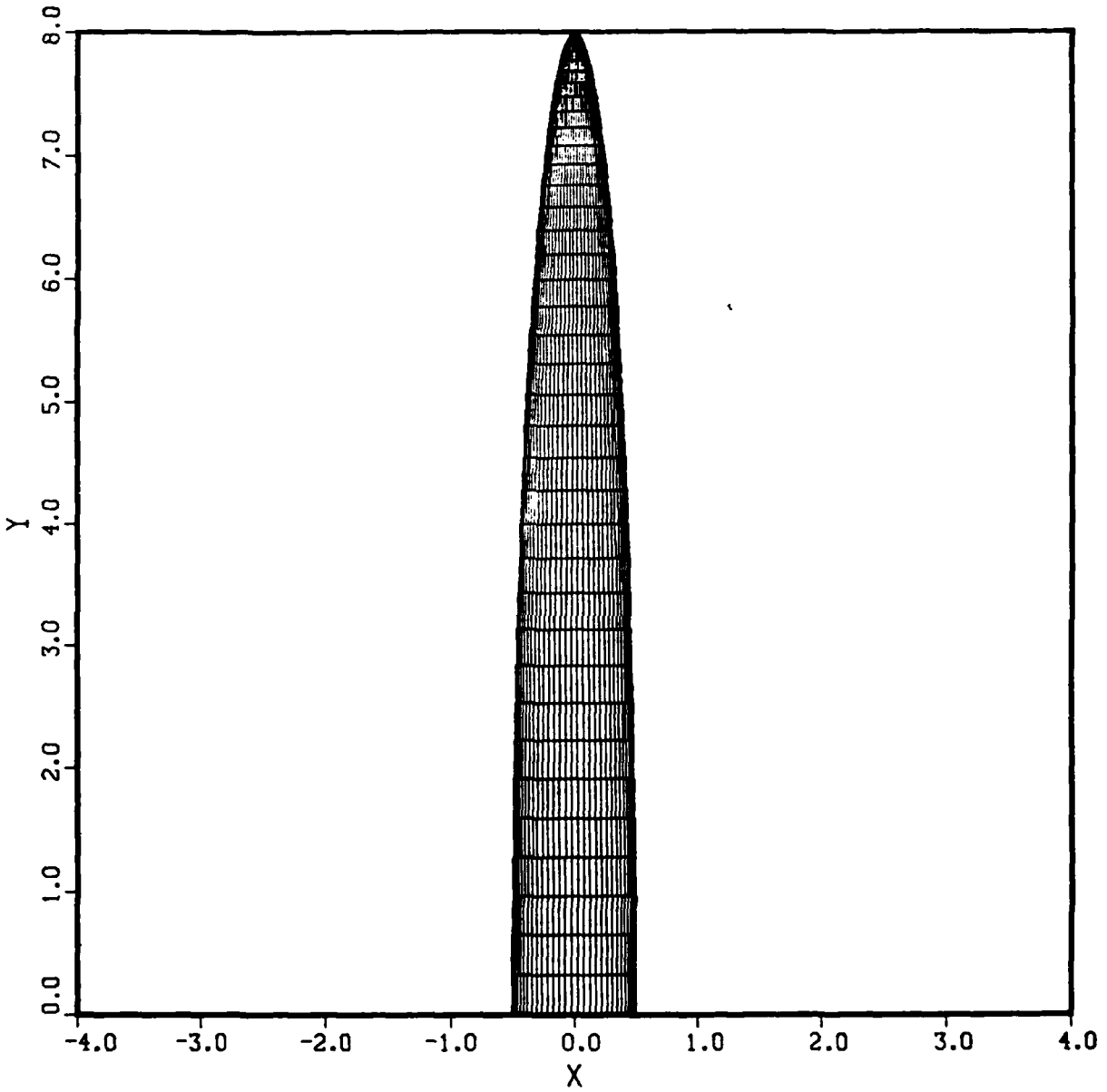


Figure 3. Surface Distribution on Wing with Elliptic Planform View at $\varsigma = 0$

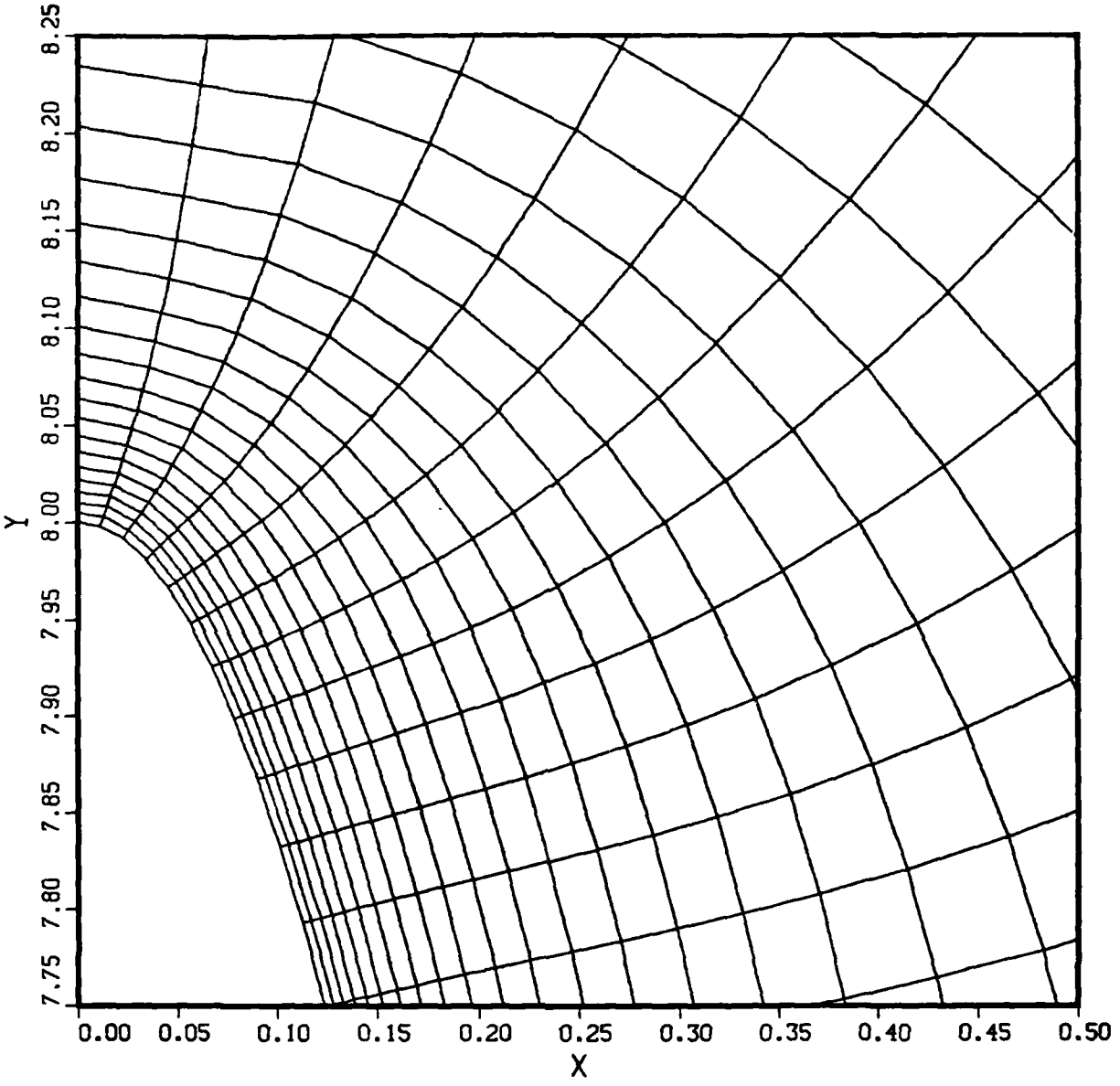


Figure 4. View of Grid Near Wing Tip, $\eta = 0$

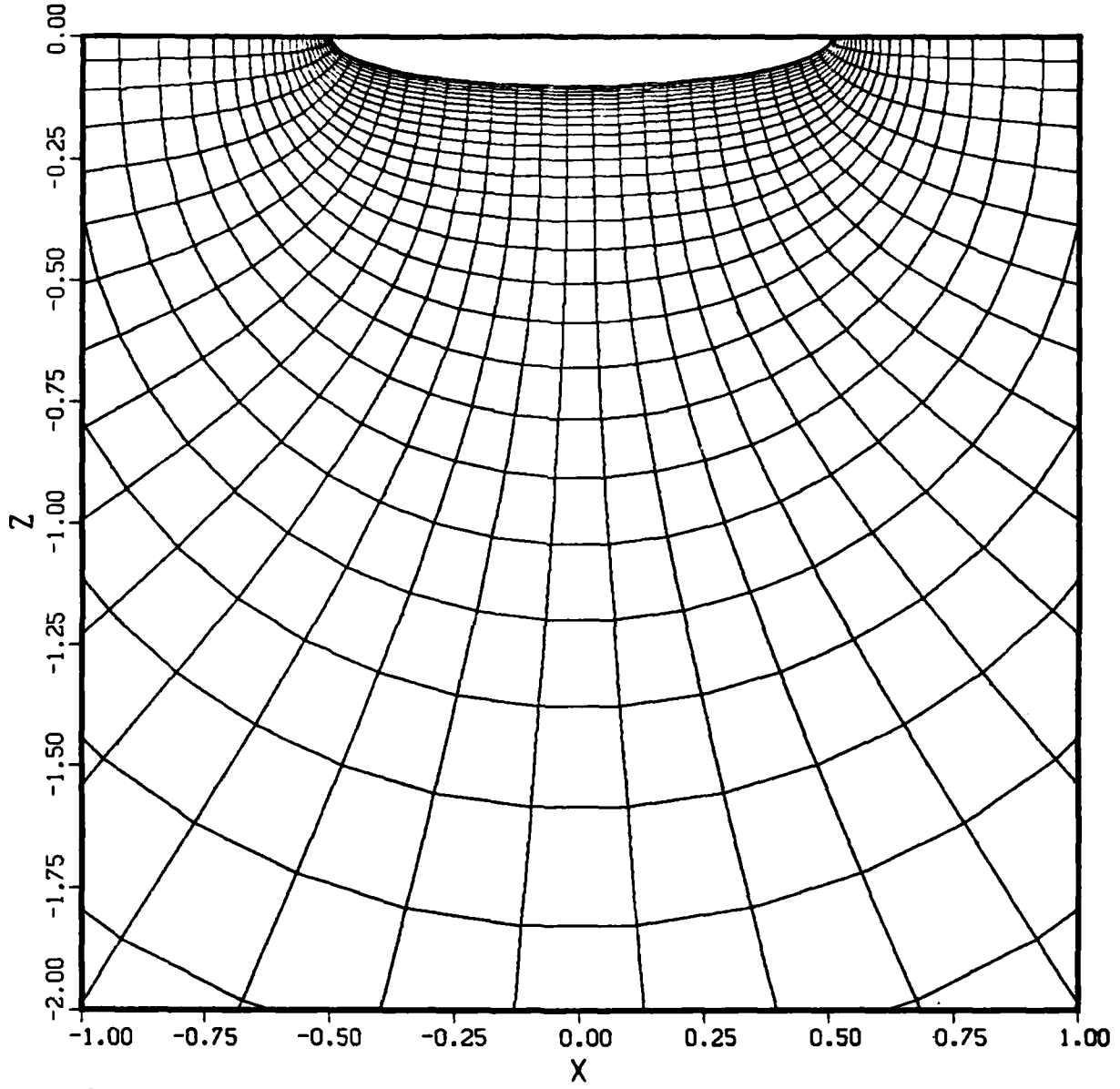


Figure 5. View of Grid at Mid-Span, Lower Surface

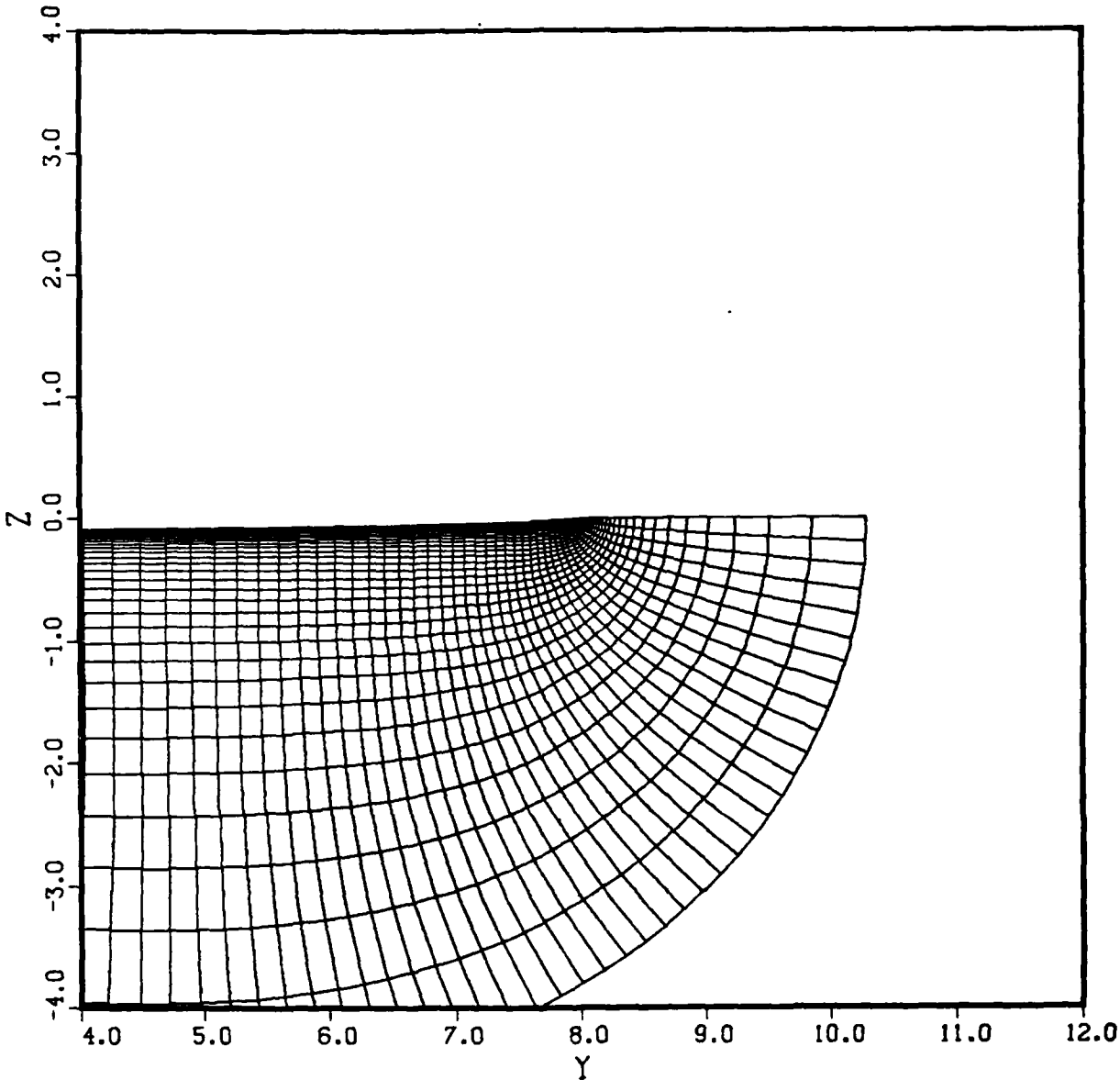


Figure 6a. View of Grid Near Tip at Lower Mid-Chord

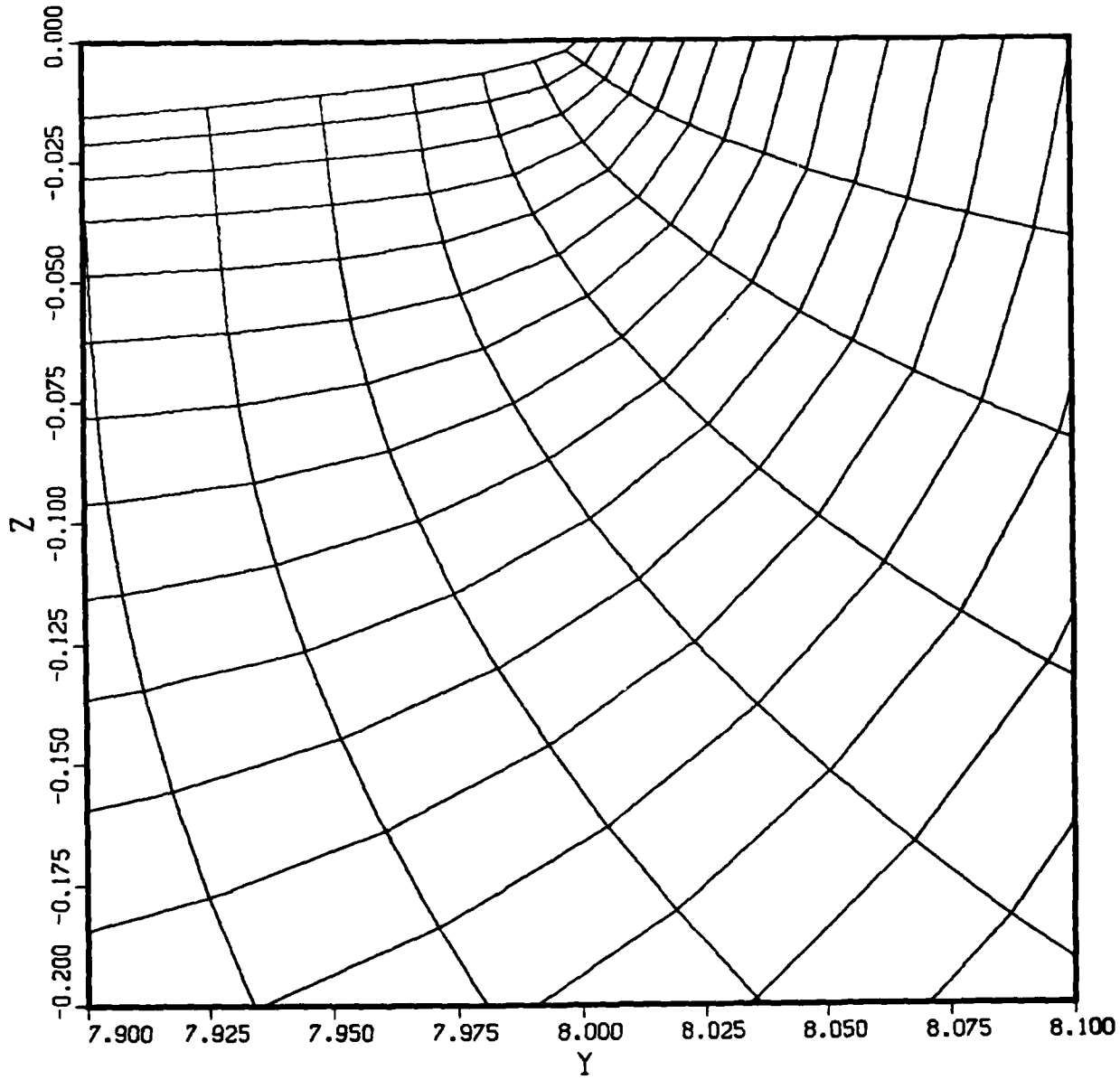


Figure 6b. Closeup of Grid Near Tip Lower Mid-Chord

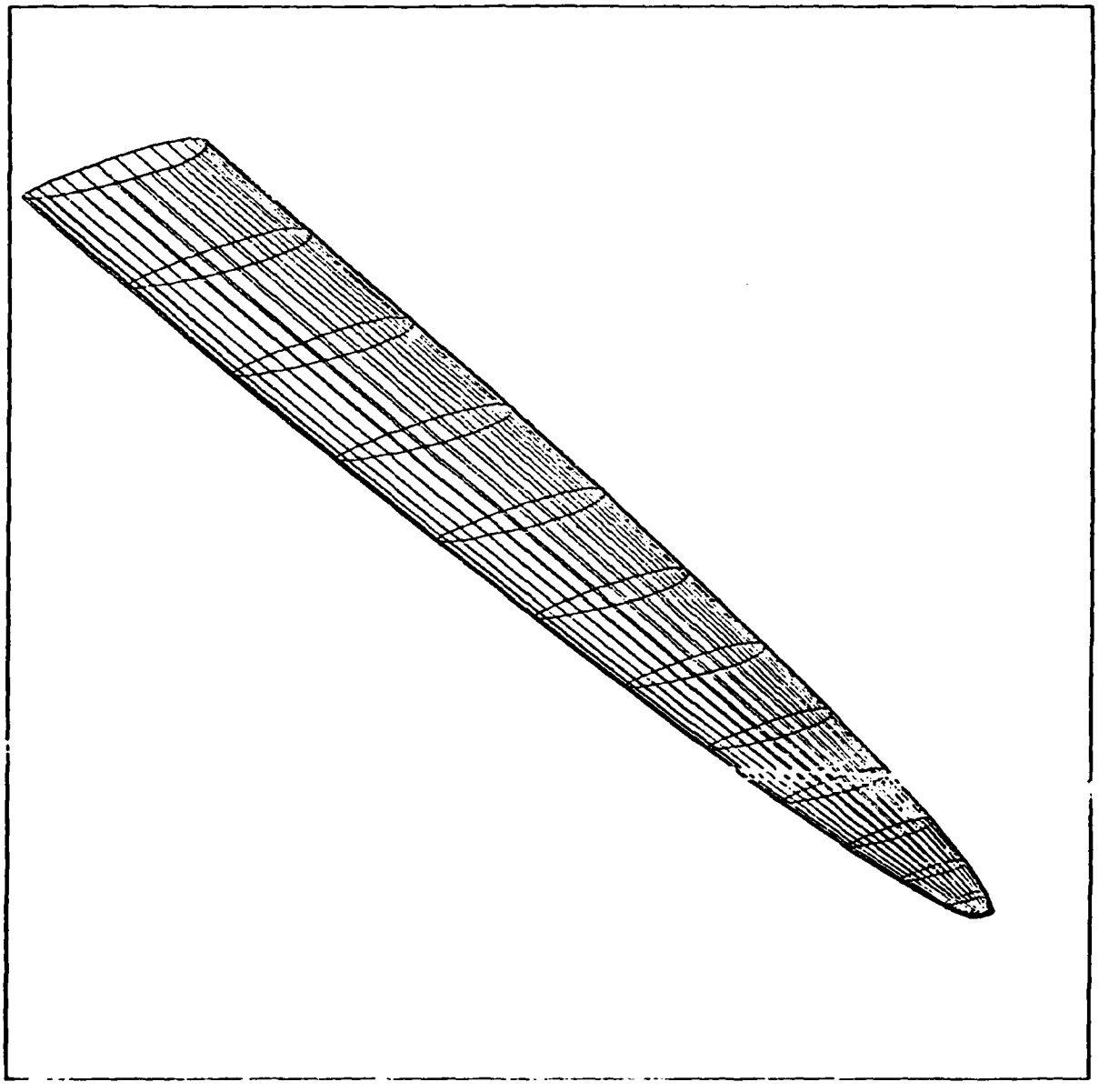


Figure 7. Overview of Wing

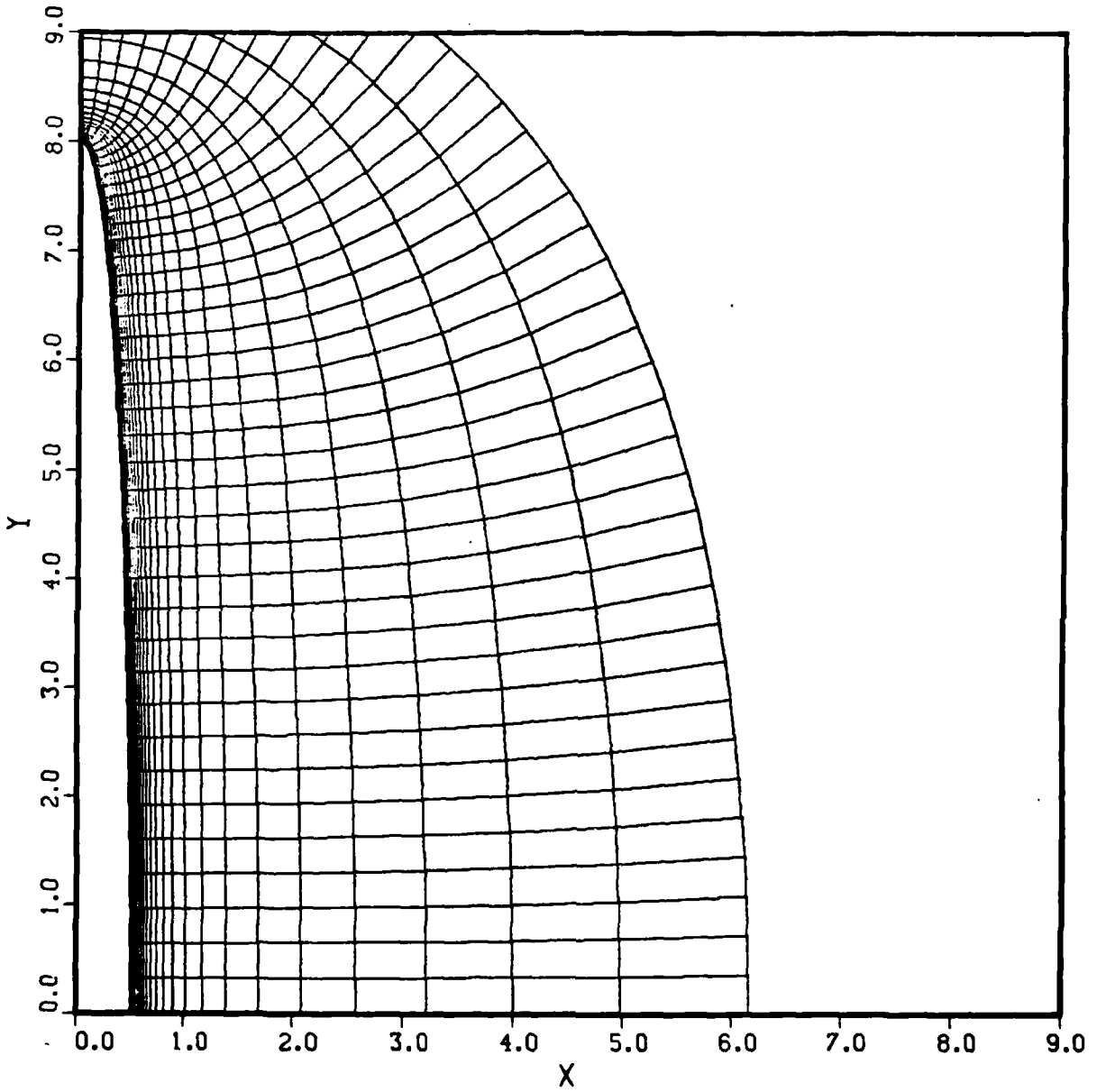


Figure 8. View of Grid Near Wing Tip, $\eta = 0$

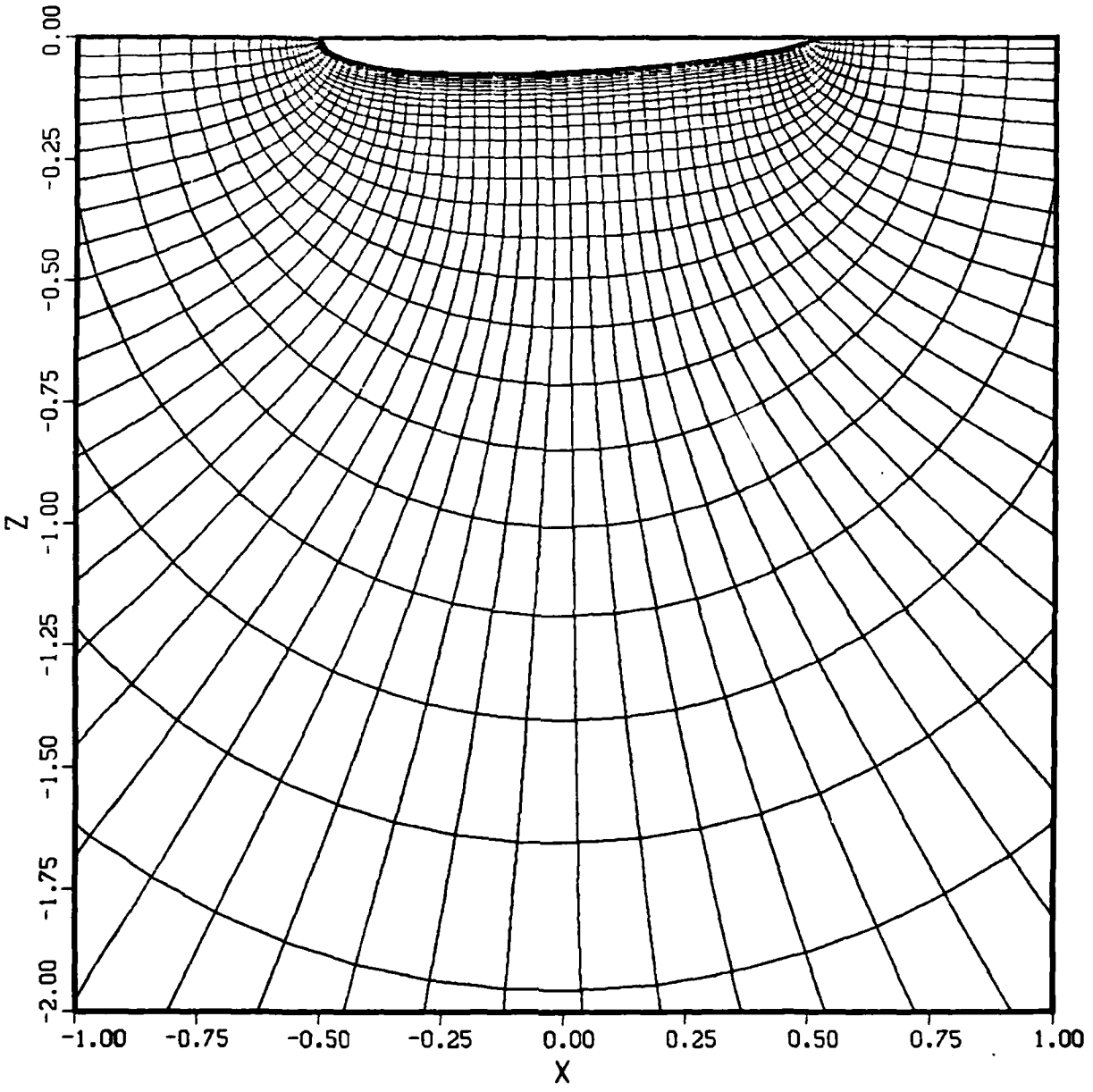


Figure 9. View of Grid at Mid-Span, Lower Surface

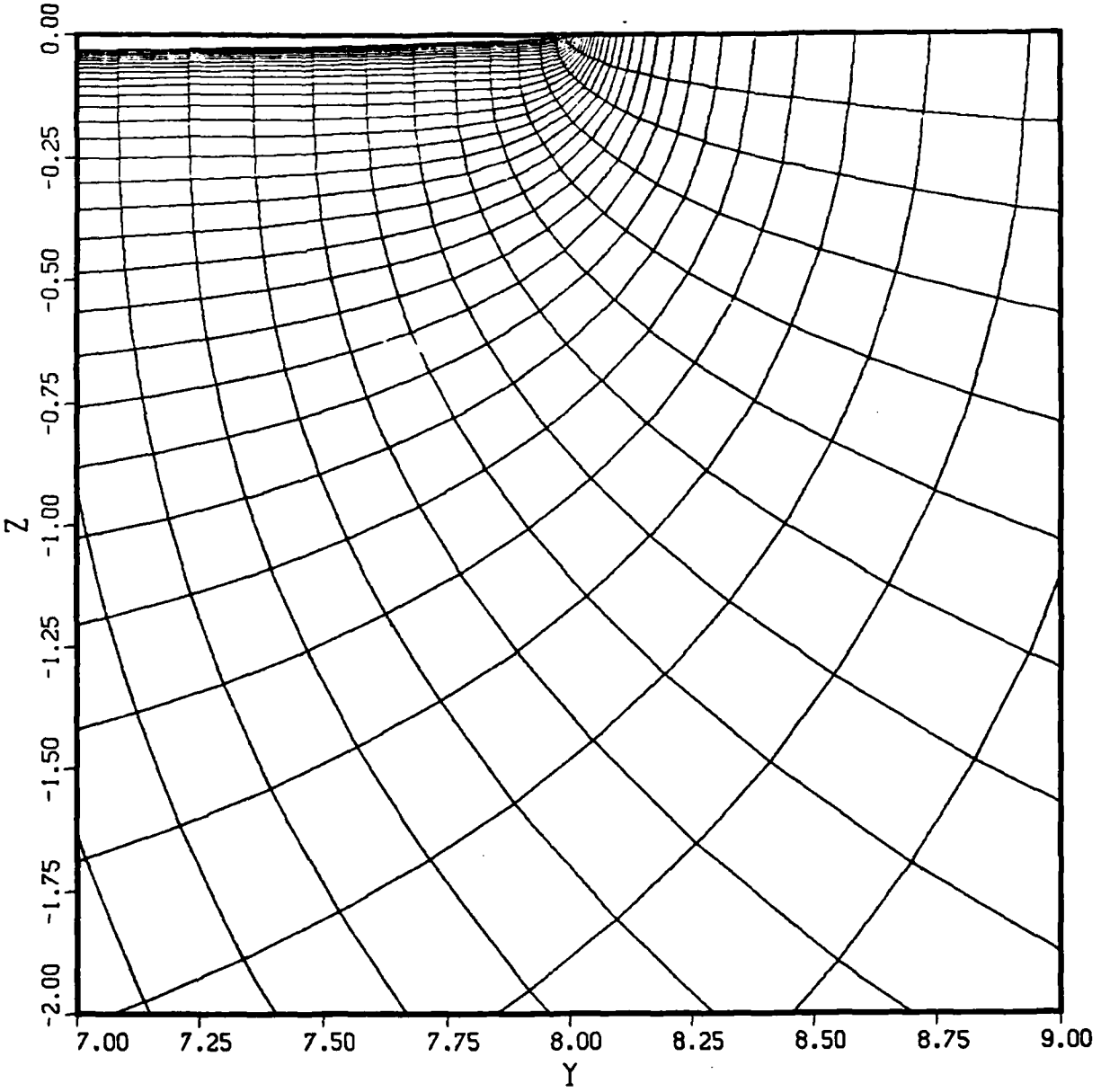


Figure 10. Closeup of Grid Near Tip Lower Mid-Chord

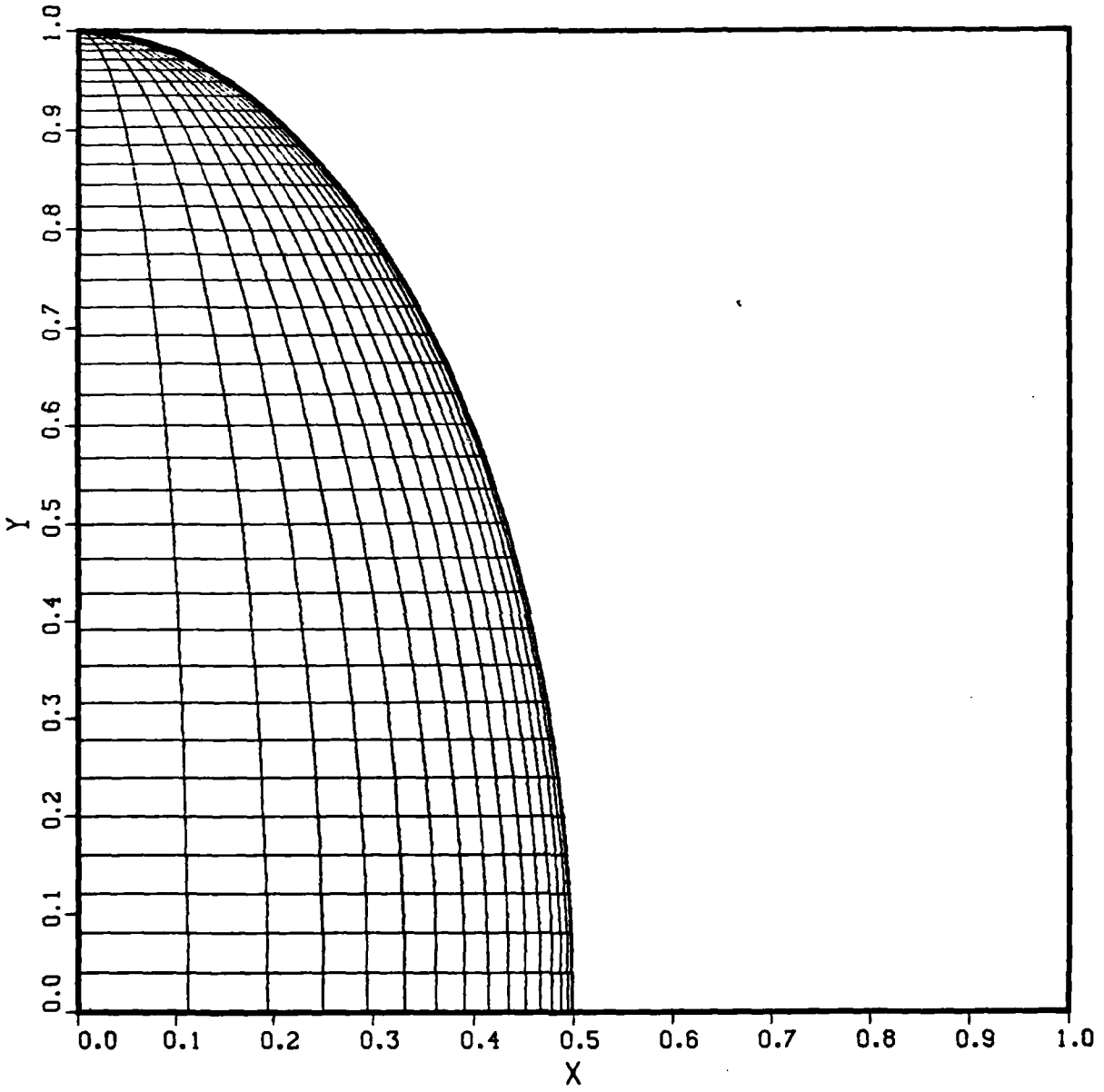


Figure 11. Surface Distribution on Bluff Body

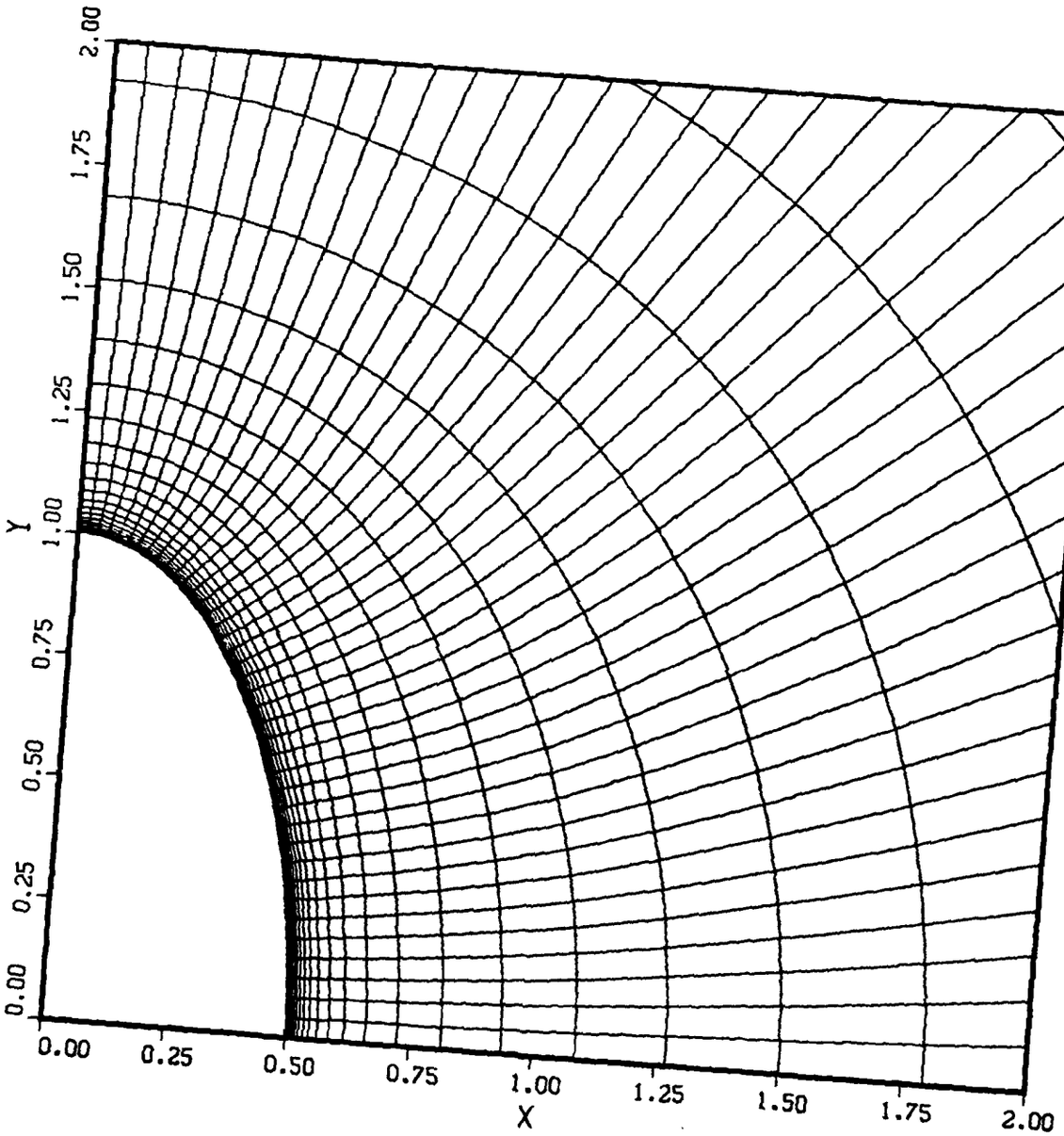


Figure 12. View of Grid Near Front Axis

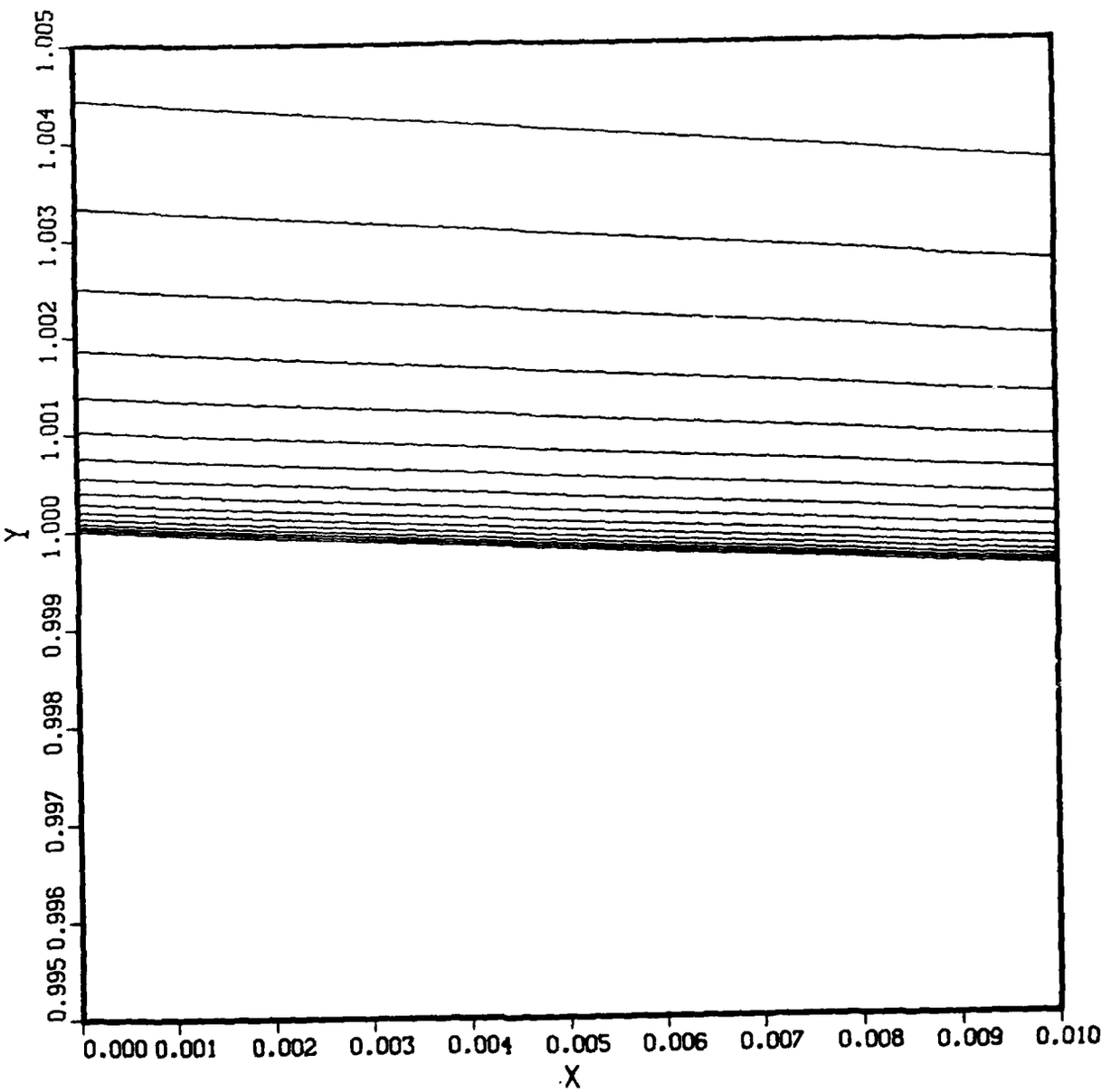


Figure 13. Closeup View of Grid Line Spacing Off Body

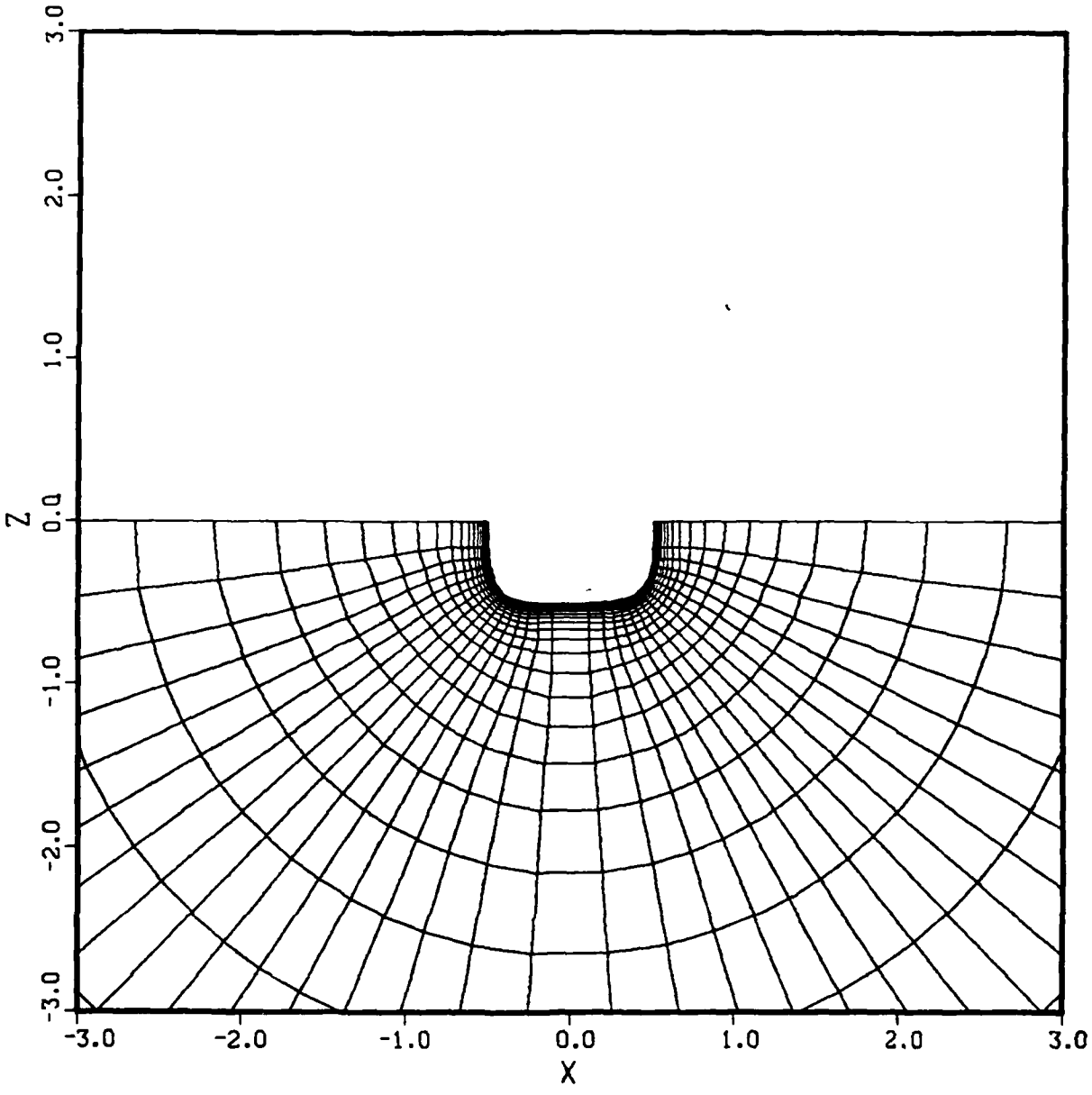


Figure 14. View of Grid with Body Cross Section

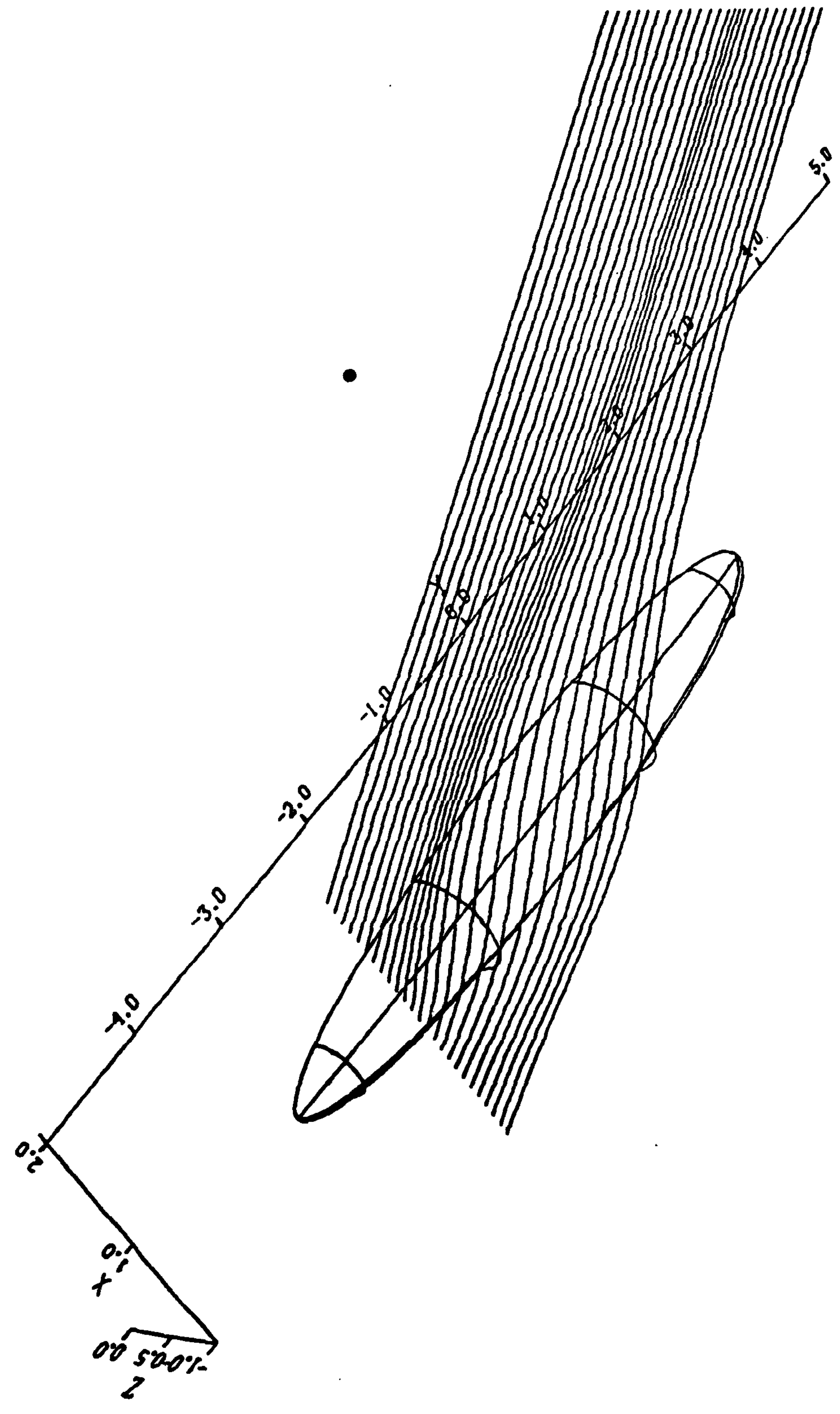


Fig 15 Particles paths about 6:1 ellipsoid, $M_\infty = .74$, $\alpha = 25°$

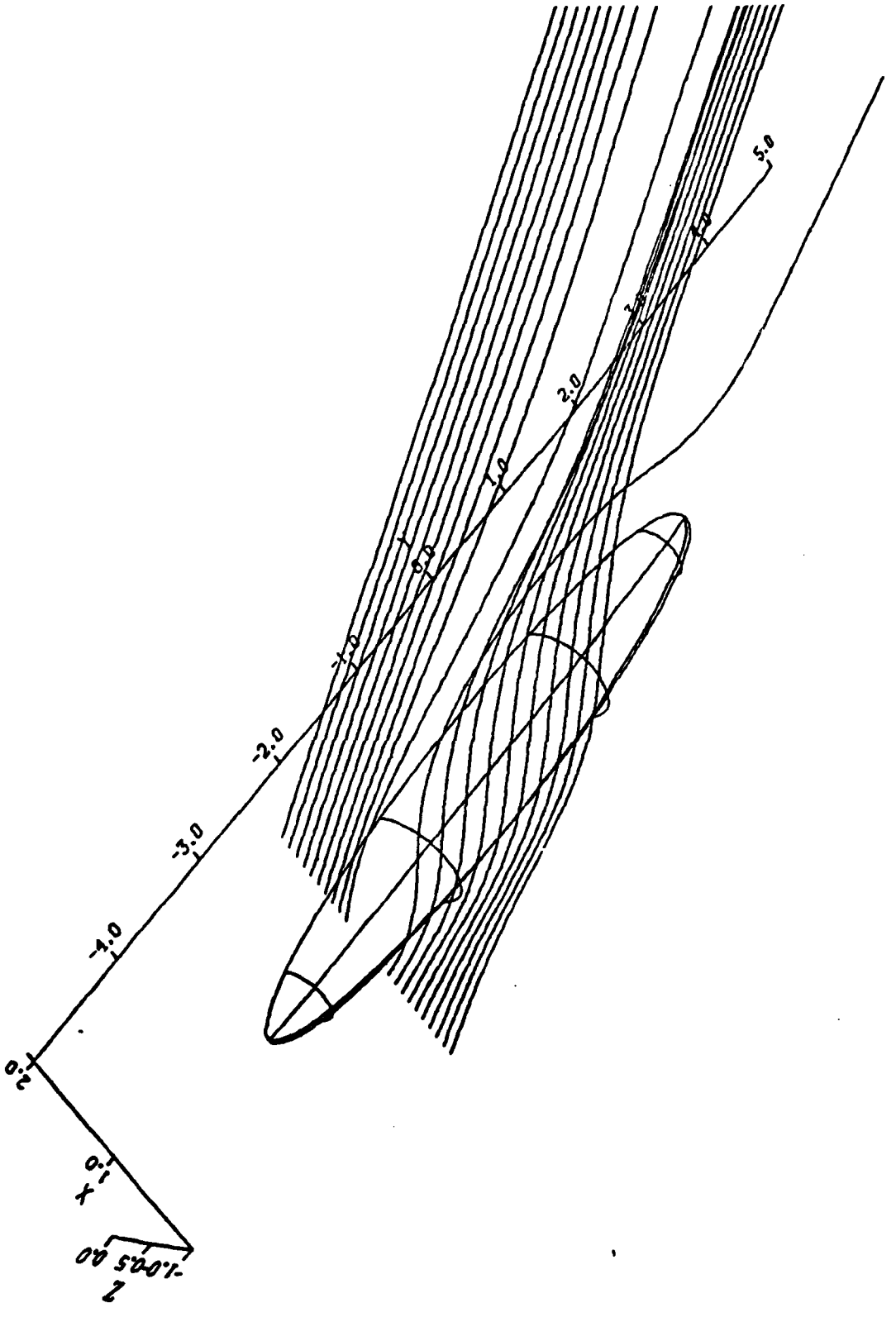


Fig. 15 continued

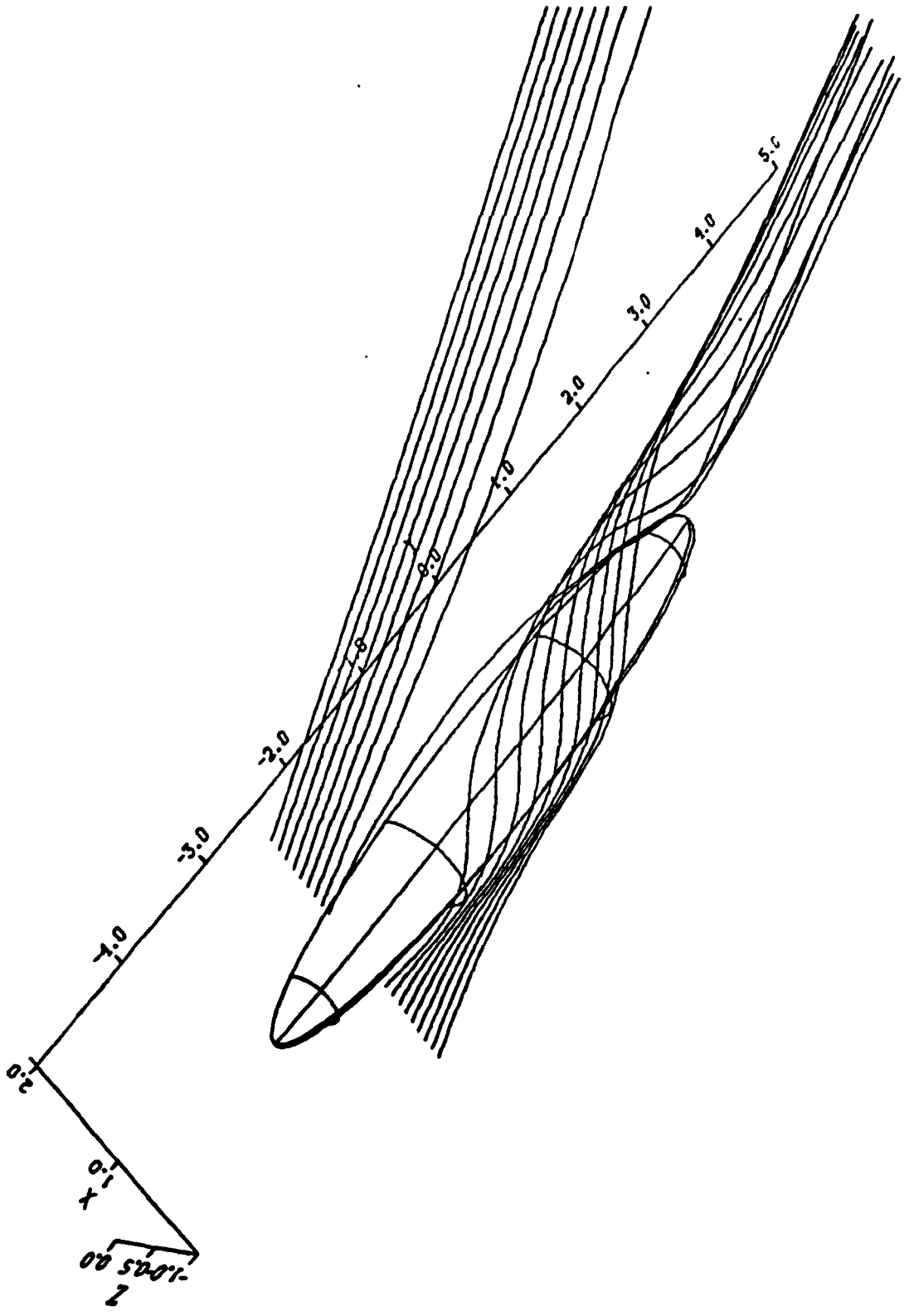


Fig. 15 continued

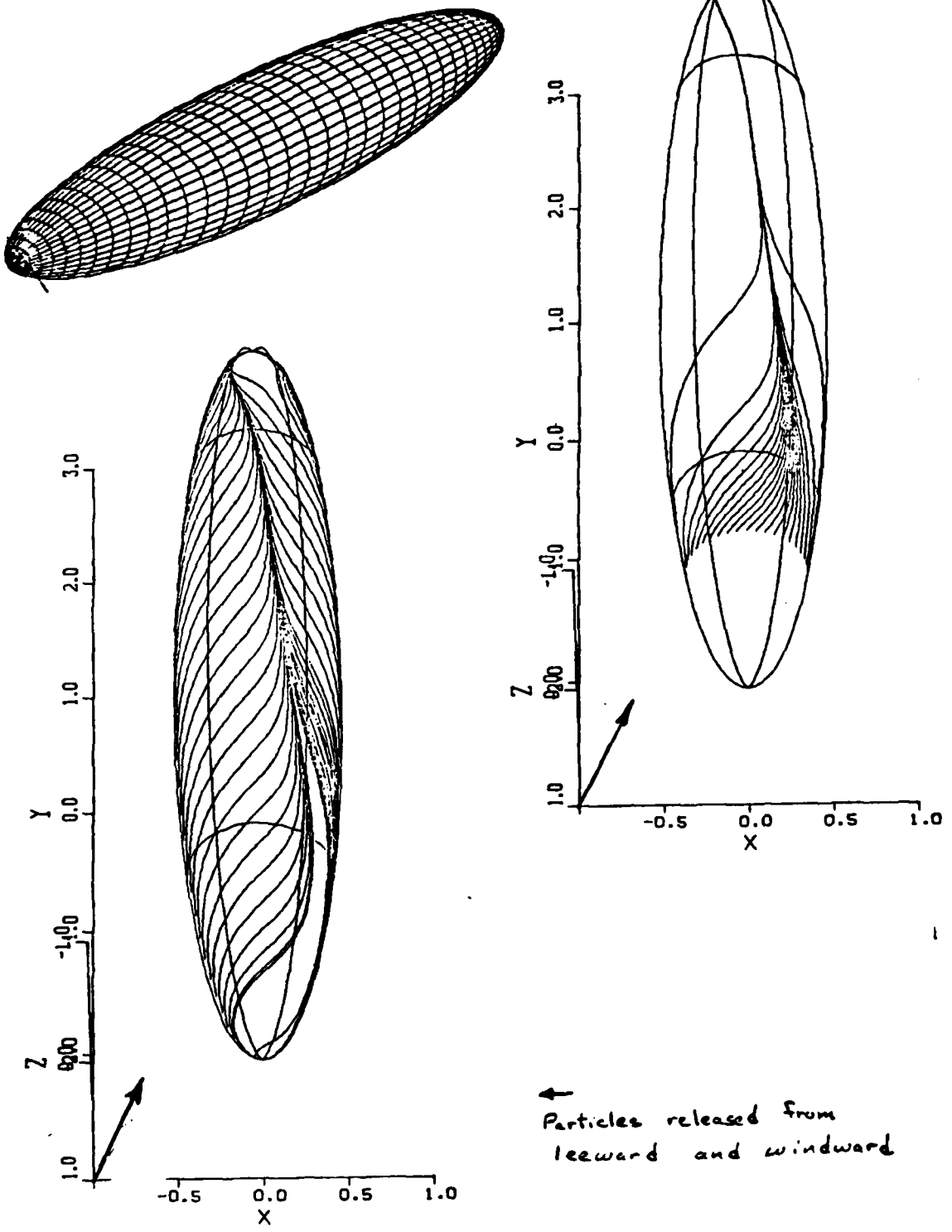


Fig. 16. Particles released along surface to simulate '0:1' flow. 6:1 ellipsoid, $Re = 44 \times 10^6$, $\alpha = 25°$, $M_\infty = .74$

PART II. EXTENSIONS TO HYPERBOLIC GRID GENERATION

In this very brief section we shall simply make mention of some problem areas and some 'fixes' in dealing with configurations with sharp edges or exceptionally concave cross sectionals. In these situations the hyperbolic grid generation scheme generally breaks down. Figure 1 indicates what breakdown can look like. Only two dimensional results are considered here with ξ the coordinate along the body and η the marching coordinate.

For exceptionally concave bodies T. J. Barth found that he could often make the hyperbolic grid generator still perform by multiplying the ξ-difference by a factor greater than one. This ad hoc fix seems to allow the generated grid lines more flexibility to adjust in ξ, yet does not seem to too adversely effect orthogonality. The success of this approach is indicated by Figures 2 to 4 which show cross sectional grid views about the X-24c which where generated for AFWAL. Barth and Risk have also generated grids about cross sections of the Shuttle, a grid is shown in Fig. 5.

The hyperbolic grid generation scheme will also break down when a grid must be wraped around a sharp edges such as the trailing edge of an airfoil meshed with an "O-grid" (i.e. warped cylindrical grid). However by treating the grid line leaving the trailing edge as a special surface at which orthogonality is not imposed, airfoil O-grids have been generated. Figures 6a and 6b show a grid about a cambered airfoil is which the ξ=constant line emanating from the trailing edge is specially treated. Near the body this line is determined so that it has a slope that bisects the trailing edge angle, while the arc length increments between points is user specified. Away from the body surface this special logic is blended into the usual grid generation formula. As the grid views indicate, such trailing edge logic can work well, but proper specification of the arc lengths along the special plane is still very much an art. We are also working on special logic that uses slope and volume information, and this is evolving into a more automatic process. A result obtained by Risk and Barth is shown in Fig.7. Some irregularity can still be obtained, but this process is becoming reliable and automatic.

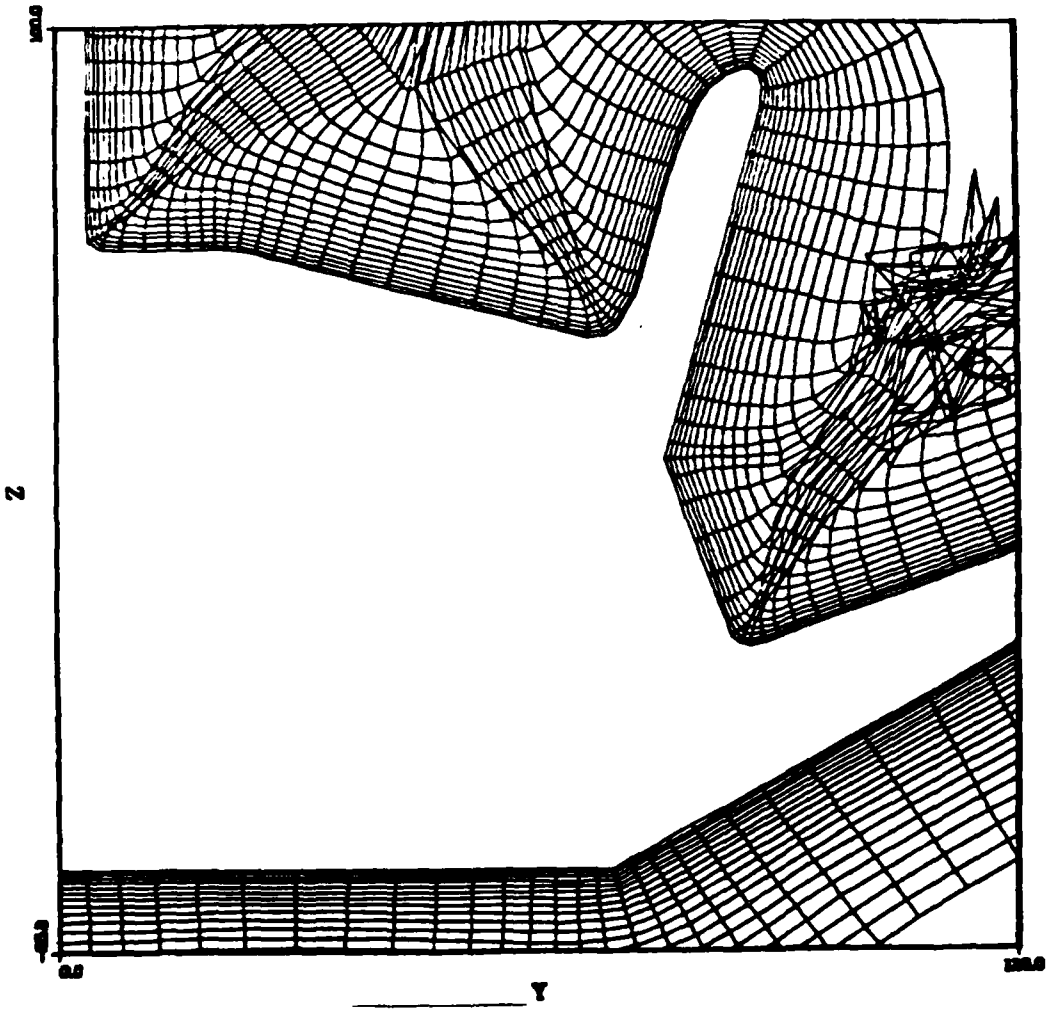


Figure 1. Example of Grid Failure

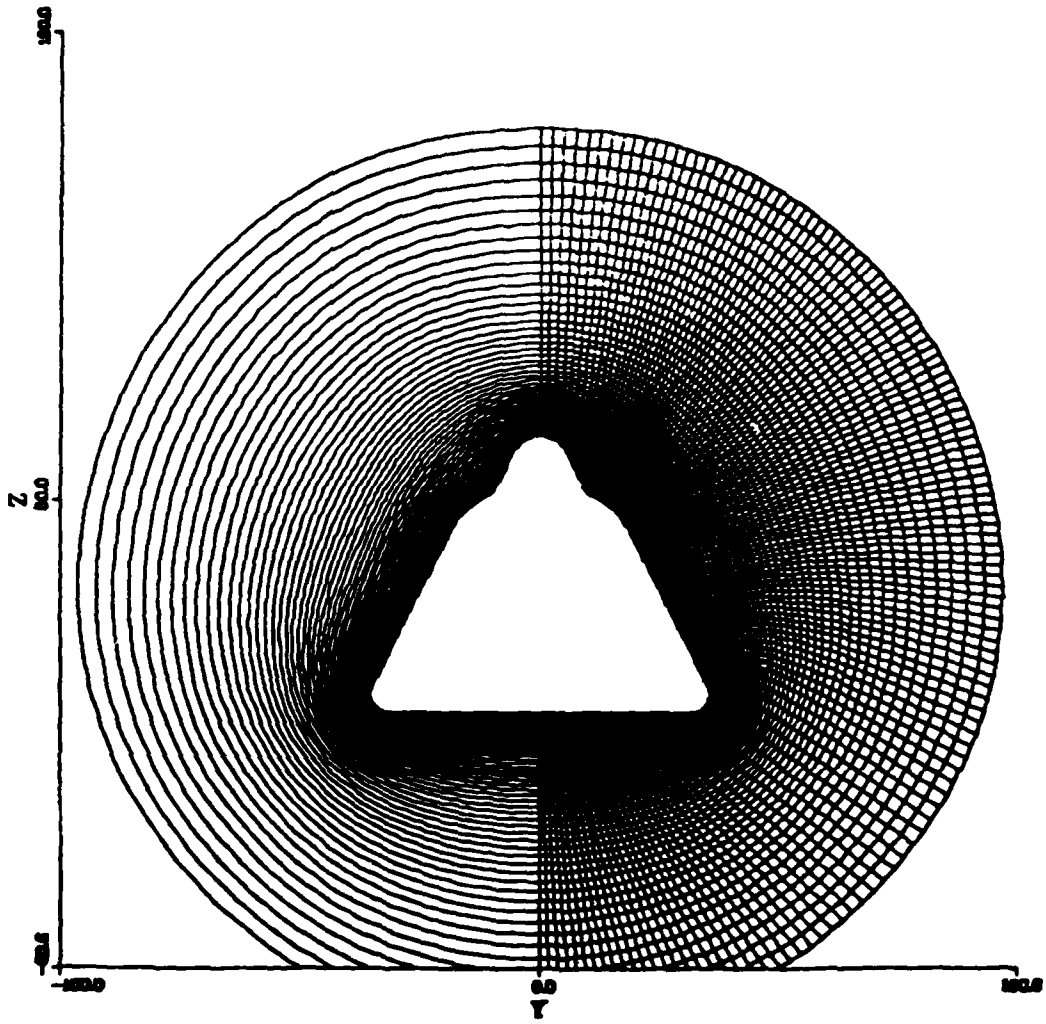


Figure 2a. Grid About X24c Cross Section

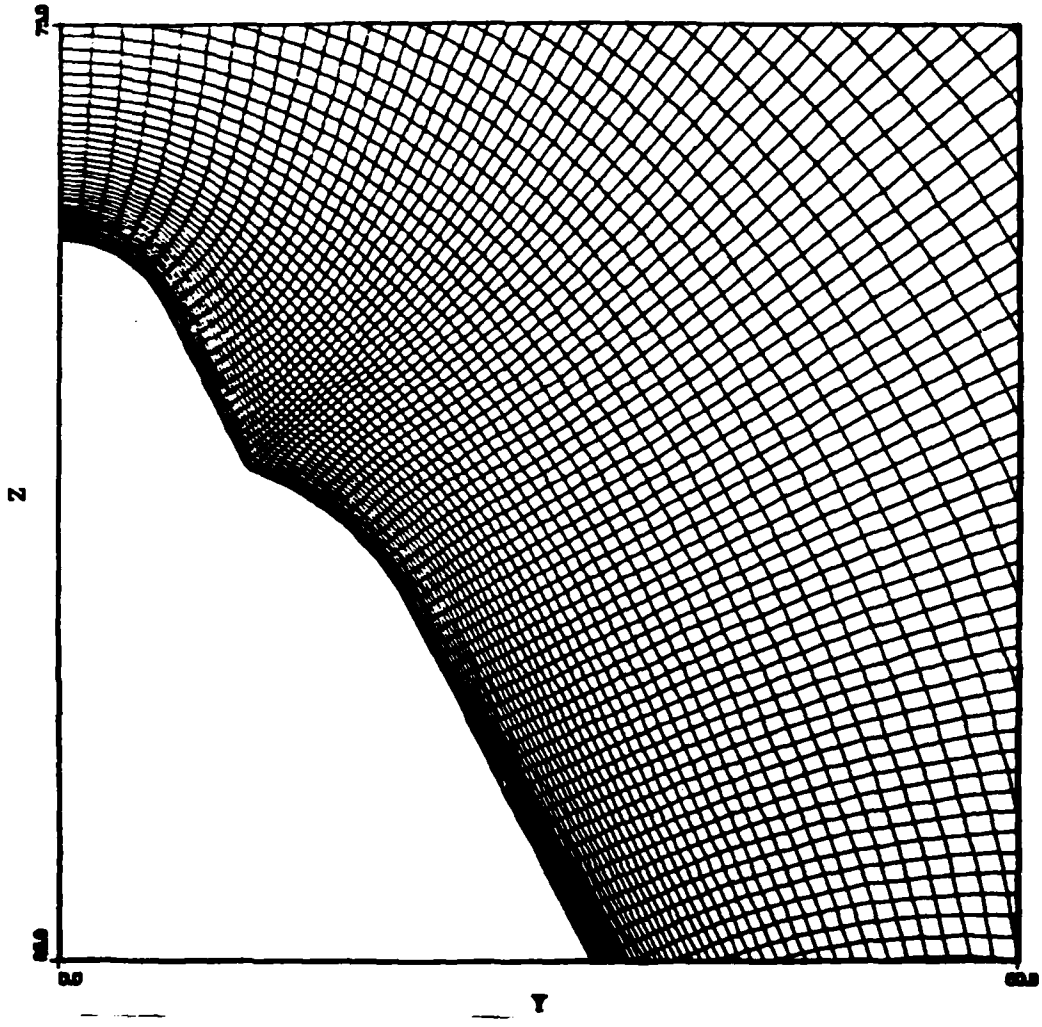


Figure 2b. Closeup of Grid About X24c Cross Section

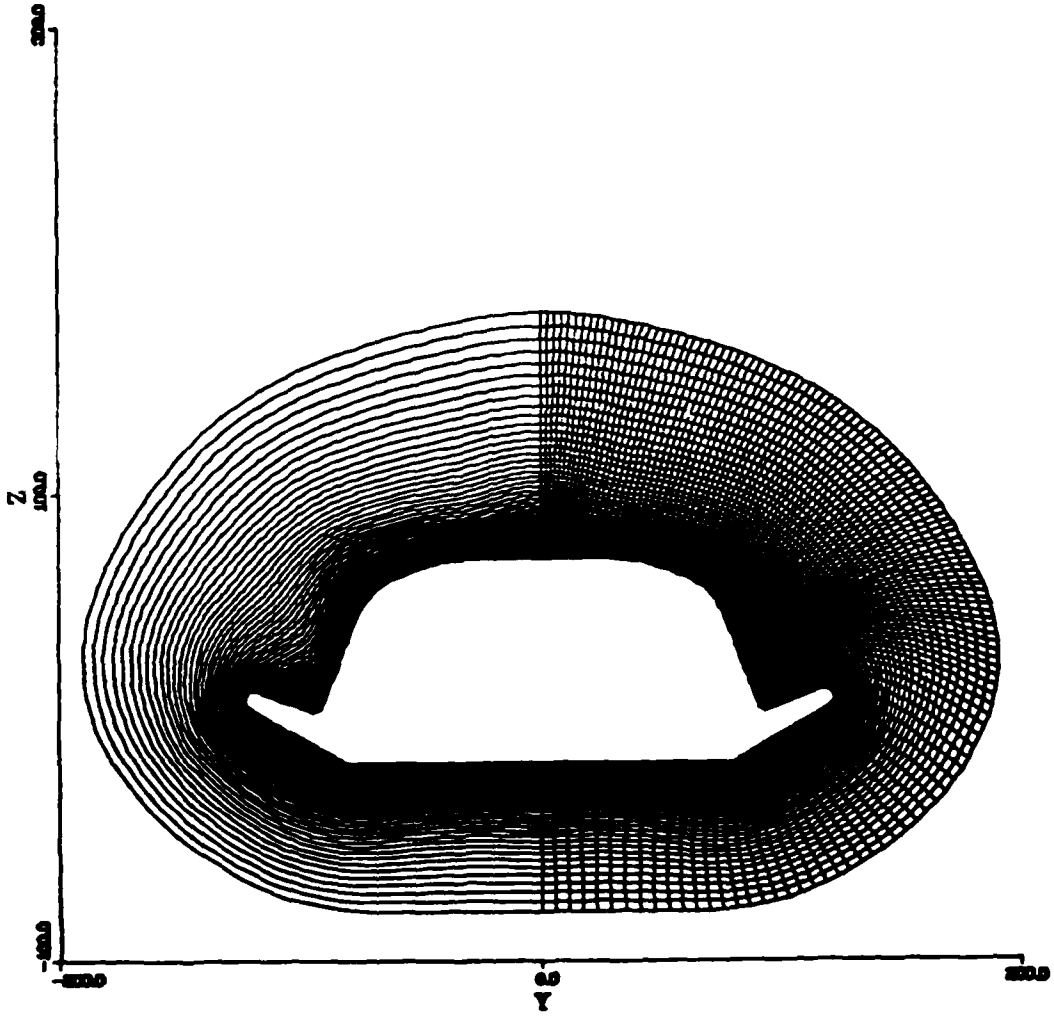


Figure 3a. Grid About X24c Cross Section

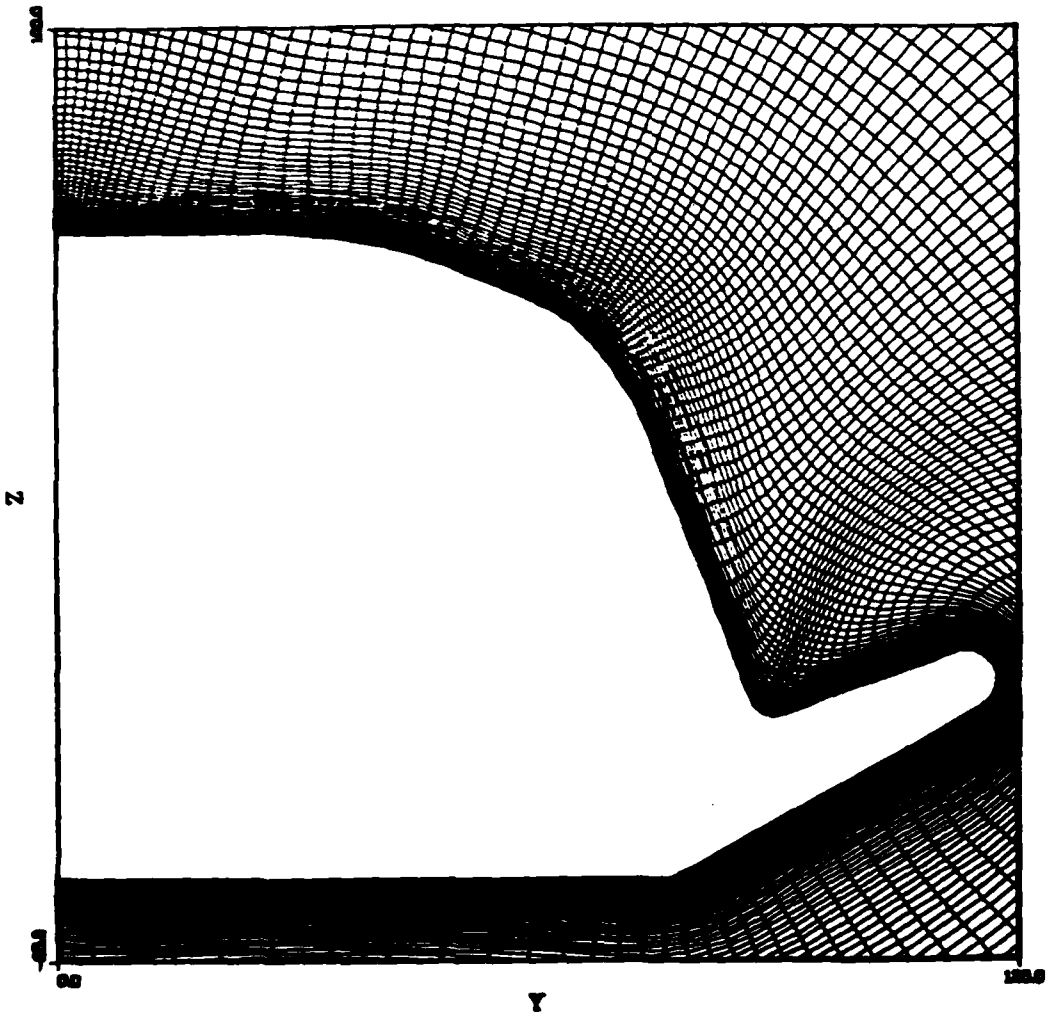


Figure 3b. Closeup of Grid About X24c Cross Section

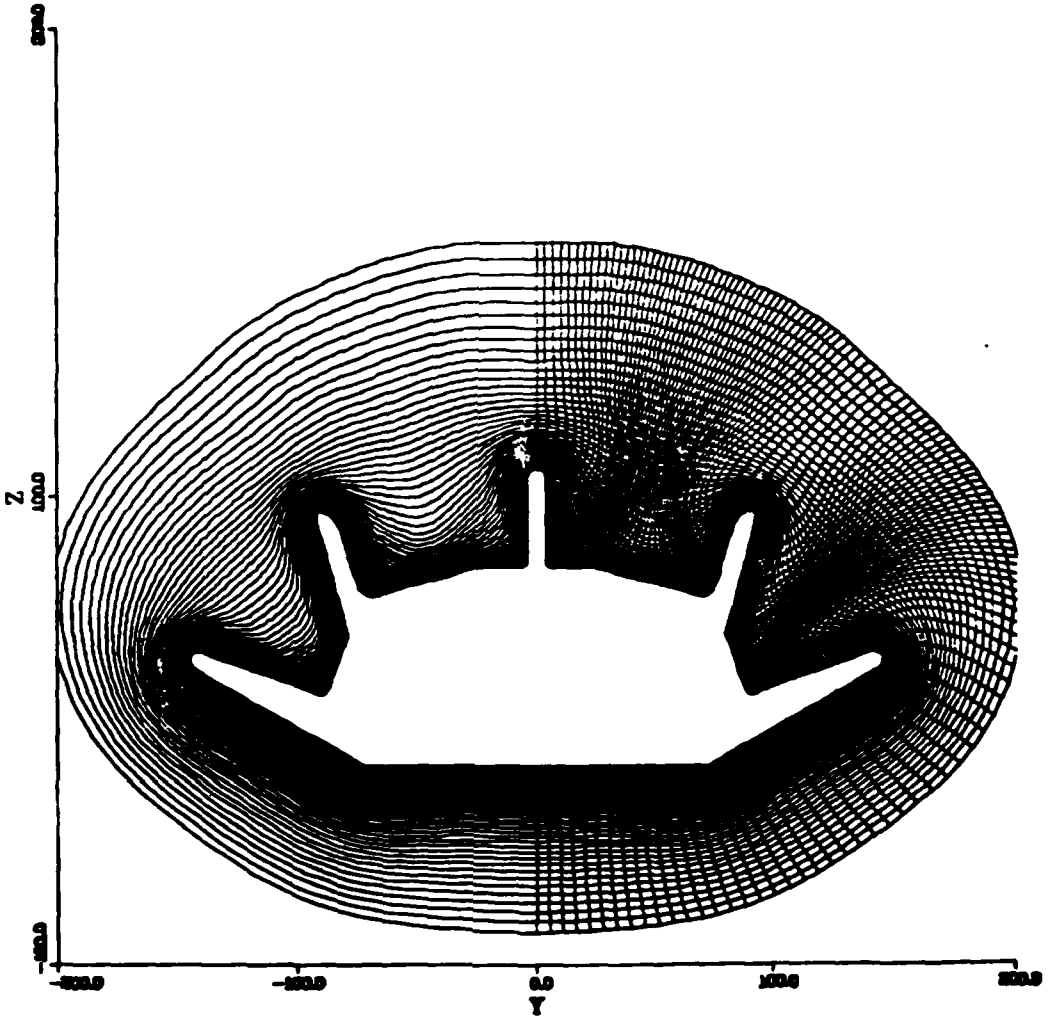


Figure 4a. Grid About X24c Cross Section

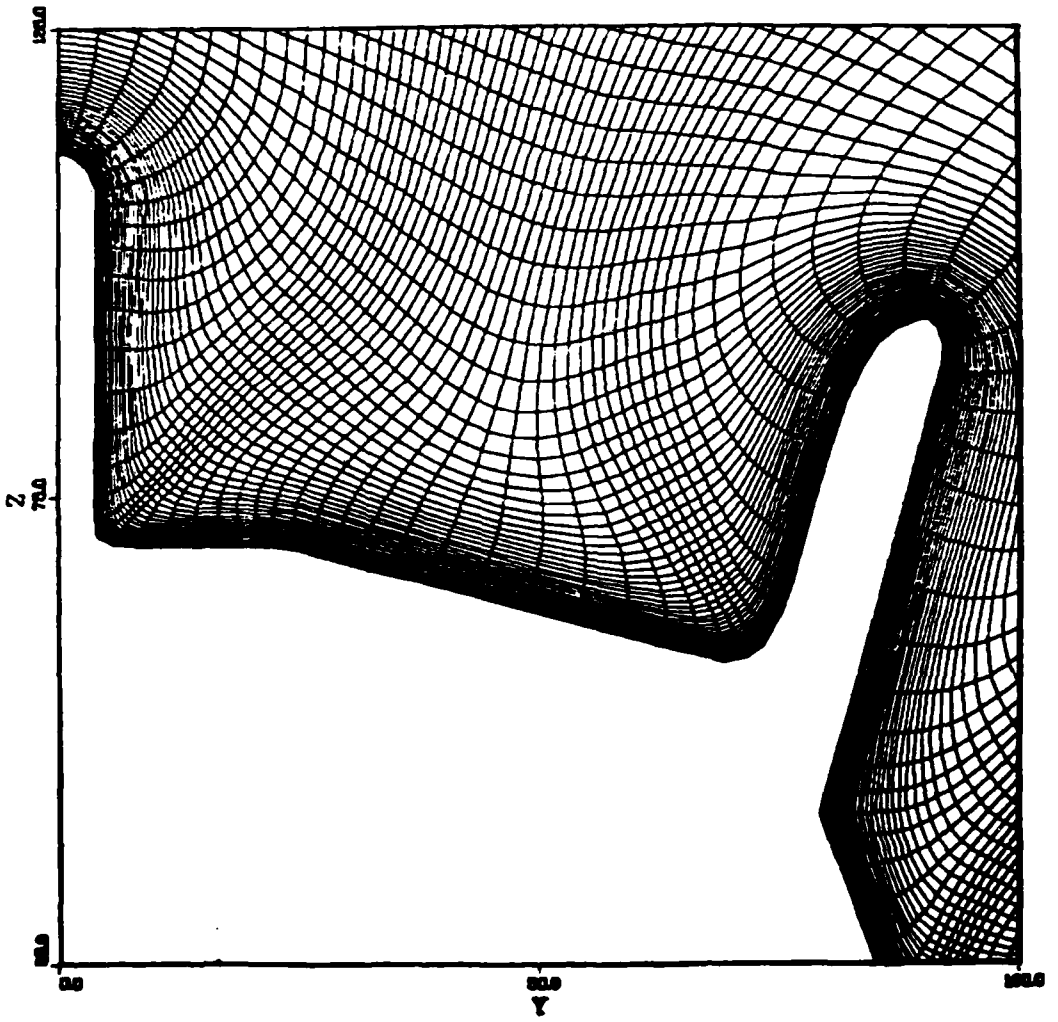


Figure 4b. Closeup of Grid About X24c Cross Section

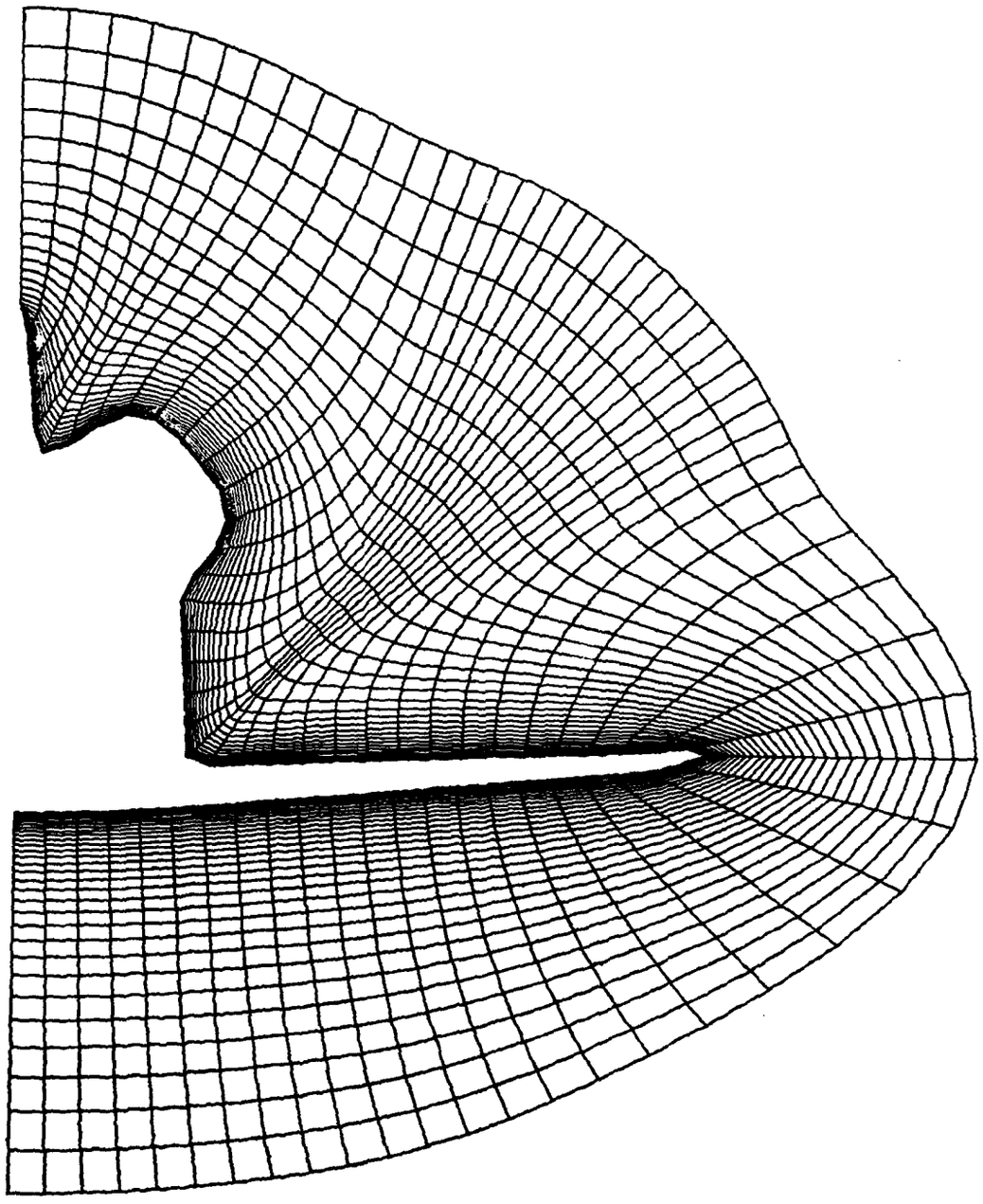


Figure 5. Grid About Shuttle Cross Section

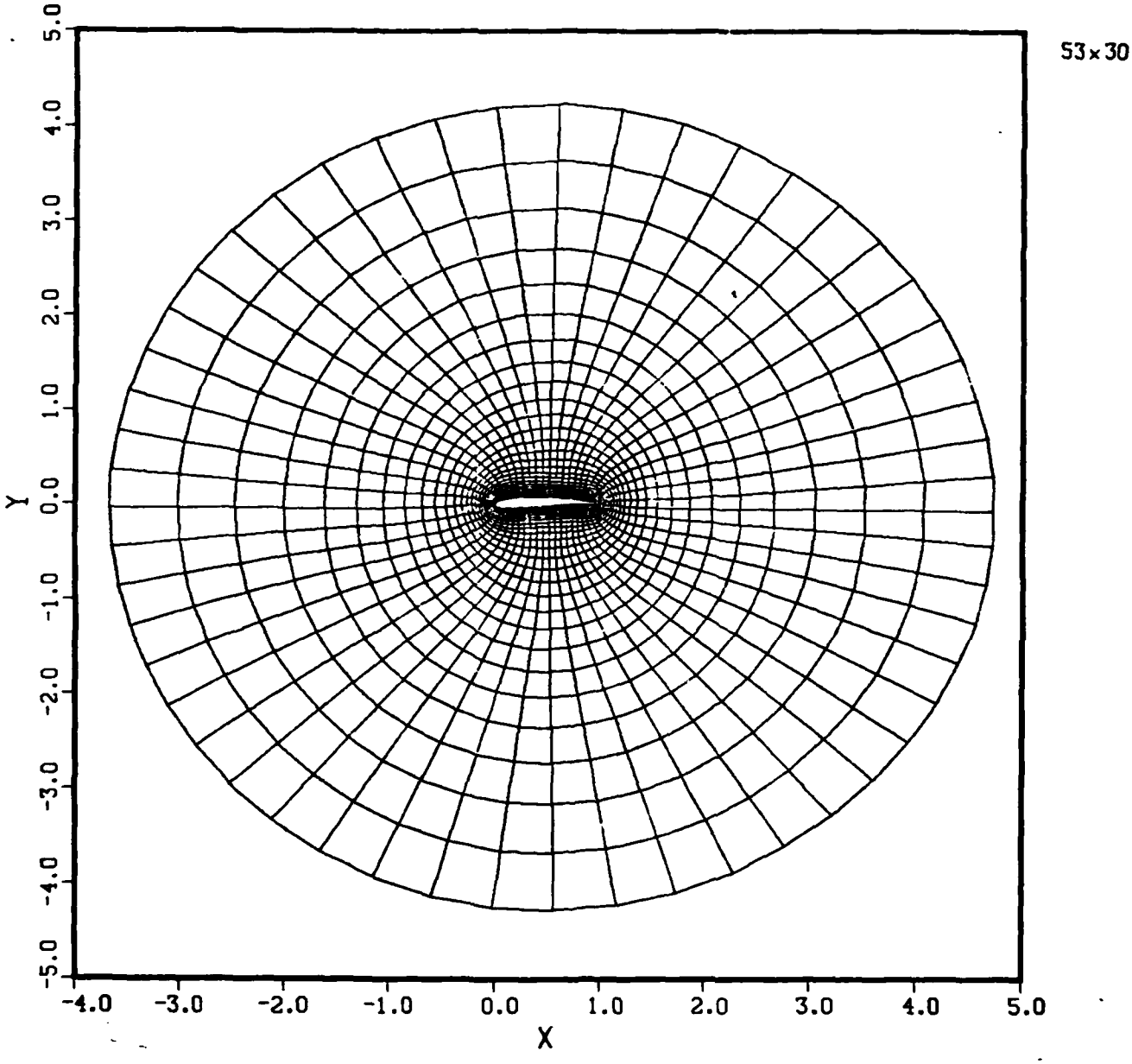


Figure 6a. Grid About Cambered Airfoil

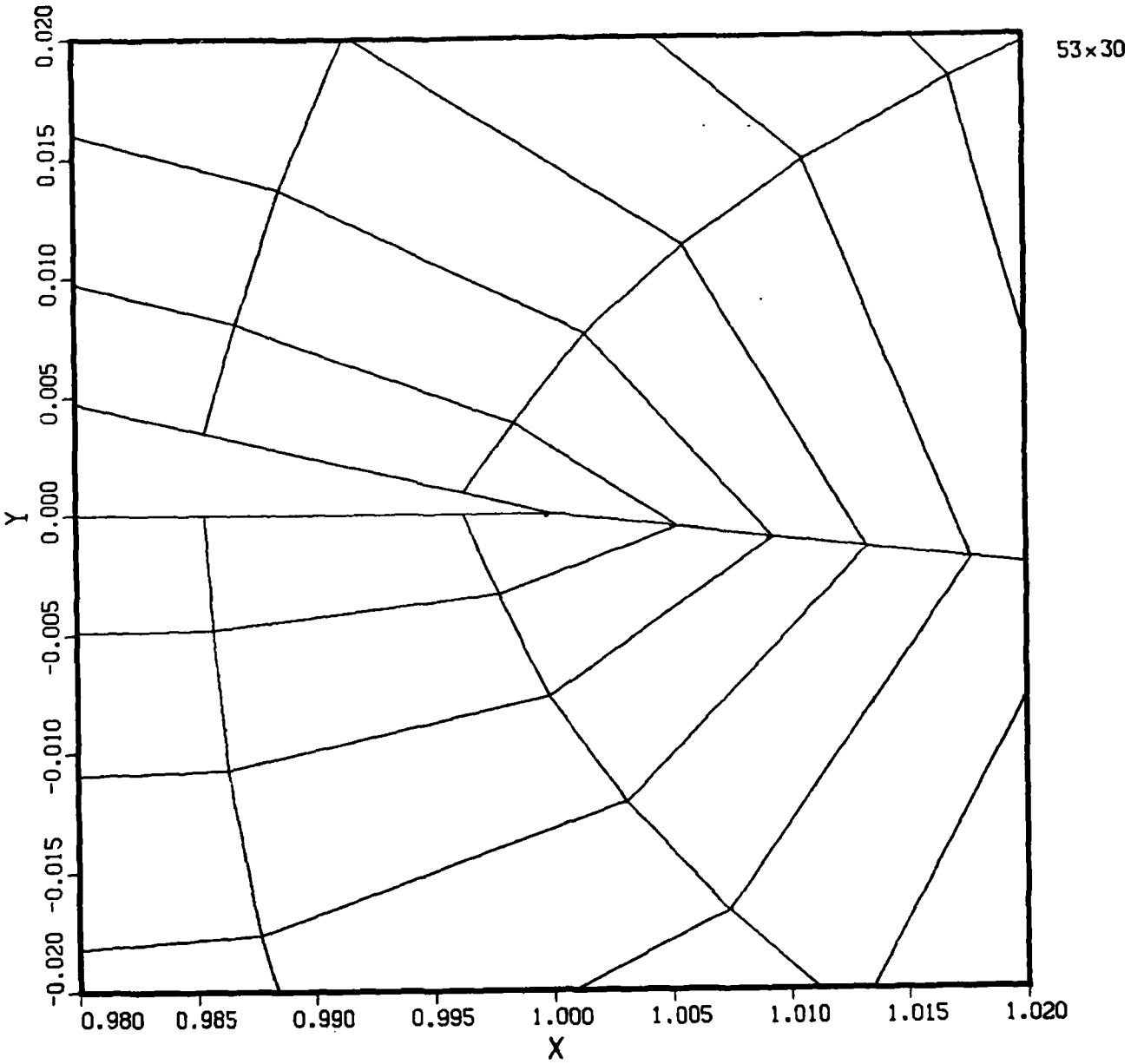


Figure 6b. Closeup of Grid Near Trailing Edge

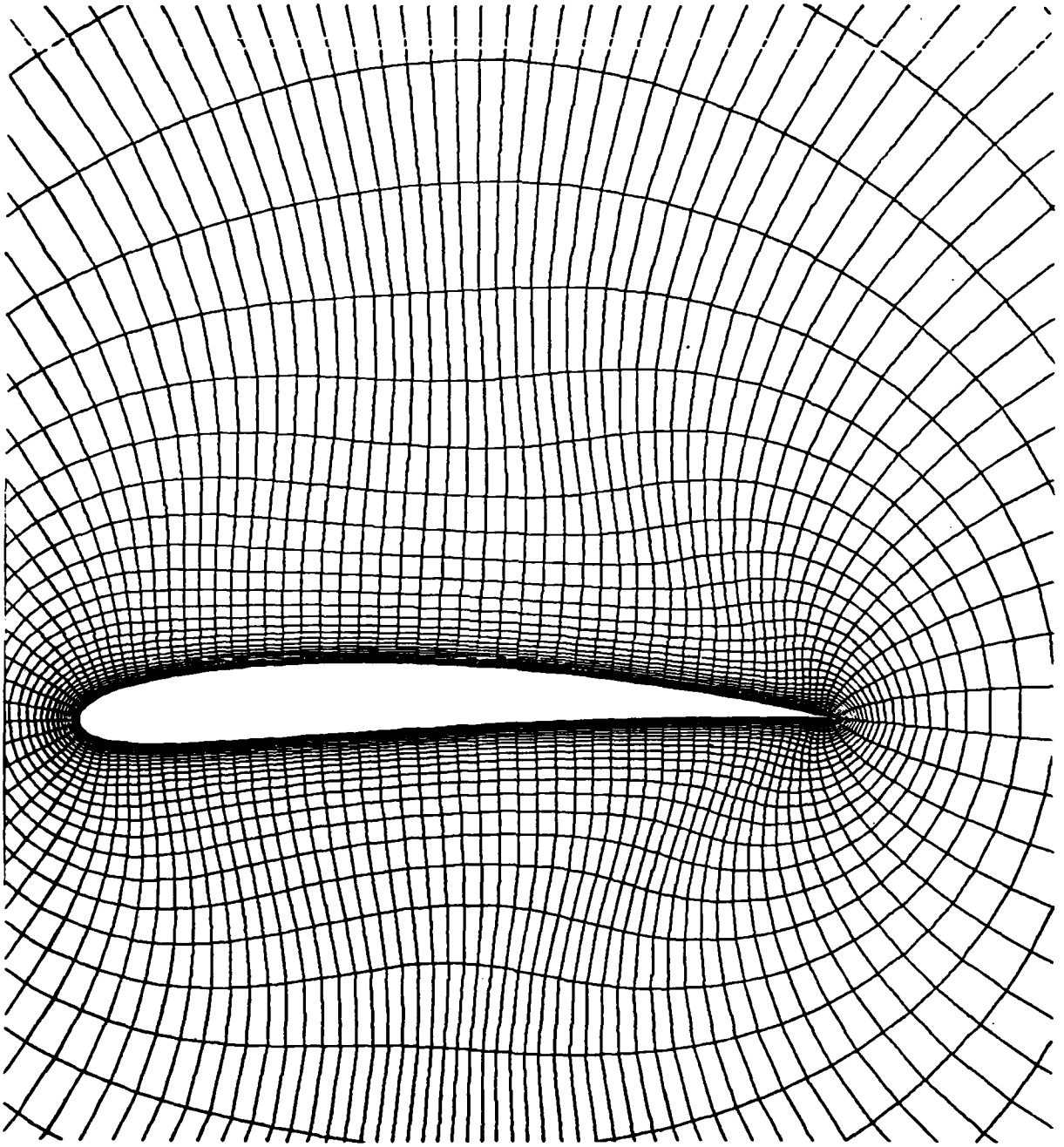


Figure 7. Grid About Cambered Airfoil

PART III. An Efficient Approximate Factorization Implicit Scheme for the Equations of Gasdynamics

TIMOTHY J. BARTH* AND JOSEPH L. STEGER†

NASA Ames Research Center, Moffett Field, CA 94035

I. Introduction

Numerical algorithms for solving the compressible Euler and Navier-Stokes equations have received considerable attention over the last decade. Both implicit and explicit time advance and iterative techniques have been ultilized. Explicit methods typically have lower arithmetic operation counts but more restrictive stability bounds than implicit methods. Use of implicit methods is generally advantageous for flow situations where an explicit time step limitation would be much less than the time step neccessary for the desired accuracy. This situation frequently arises in the solution of unsteady flows that are characterized by low frequency motion, boundary condition forcing, and high Reynolds number viscous effects. Implicit algorithms can often be readily adapted to be efficient relaxation schemes for steady state applications as well.

The purpose of this paper is to present a technique that substantially reduces the arithmetic operation count, but otherwise retains the stability characteristics of a Beam-Warming approximate factorization implicit algorithm for the Euler equations. This new method extends a matrix splitting of Steger[1], previously restricted to Cartesian coordinates, such that equation decoupling is achieved for the conservative form of the Euler equations. The equation decoupling (or matrix reducibility) results in a significant improvement in the efficiency of the algorithm. By ultilizing a local transformation, the equation decoupling techniques are extended to the Euler equations in generalized coordinates. Computation of transonic flow about a NACA 0012 airfoil is used to verify that no numerical stability is lost with the new

*Informatics General Corporation.
†Senior Staff Scientist

reducible implicit algorithm. Extension of the technique to the Navier-Stokes equations is also indicated in the Appendix A.

II. Background

In this section we review a sound speed and velocity splitting concept for an approximate factorization implicit algorithm in Cartesian coordinates and describe how it can be used to improve computational efficiency. The extension for general coordinate systems is described in the following section.

a) Beam and Warming Algorithm

The conservative form of the Euler equations for a perfect gas can be written in Cartesian coordinates as

$$\partial_t Q + \partial_x E + \partial_y F = 0 \tag{2.1}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho v u \\ u(e+p) \end{bmatrix}, \quad F = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ v(e+p) \end{bmatrix} \tag{2.2}$$

and

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right) \tag{2.3}$$

Here ρ is the fluid density, u and v are the Cartesian velocity components, e is the total energy per volume, and p is the pressure.

A general purpose implicit finite difference scheme (c.f. Refs. 2 and 3) for equation (2.1) that can be readily adapted to either steady or unsteady flow applications is given by

$$\begin{aligned} \left[I + h\delta_x A^n + D_x^{(2)}\right]\left[I + h\delta_y B^n + D_y^{(2)}\right]\Delta Q^n = \\ -\Delta t\left[\delta_x E^n + \delta_y F^n + D^{(4)} Q^n\right] \end{aligned} \tag{2.4}$$

where

$$D^{(4)} = \epsilon_e \Delta t[(\nabla\Delta)_x^2 + (\nabla\Delta)_y^2] \tag{2.5a}$$

and

$$D_x^{(2)} = -\epsilon_i \Delta t (\nabla \Delta)_x, \quad D_y^{(2)} = -\epsilon_i \Delta t (\nabla \Delta)_y \tag{2.5b}$$

Here δ_x and δ_y are three-point, second order accurate central space difference operators, while Δ and ∇ denote forward and backward space differences. Either first or second order accurate time differencing may be used with h=Δt or $\Delta t/2$, alternately the time step may be used as a relaxation parameter and may vary in space as well. With the use of central space differencing numerical dissipation is needed and implemented with the second and fourth order differences given above. The parameters ϵ_e and ϵ_i control the amount of numerical dissipation. These parameters may be constants or may vary with the solution gradients (in which case they are moved inside the operators so as to maintain conservative form). Steady state convergence can be greatly enhanced by more implicit treatment of the dissipation terms and by use of space varying scaling parameters.

In equation (2.4), local time linearizations have been ultilized to avoid solving nonlinear equations between time or iterative levels n to n+1 by expanding E and F in terms of ΔQ as

$$E^{n+1} = E^n + A^n(Q^{n+1} - Q^n), \quad A = \partial E/\partial Q \tag{2.6a}$$

$$F^{n+1} = F^n + B^n(Q^{n+1} - Q^n), \quad B = \partial F/\partial Q \tag{2.6b}$$

The left hand side operators of equation (2.4) form a linear system of equations that must be solved at each time or iteration level. Although the matrix band structure of this system has been altered by approximately factoring into one-dimensional-like operators, the inversion process has been simplified. The approximate factorization (AF) reduces the inversion work but sometimes at the cost of limiting the time step Δt because of either time accuracy considerations or because of reduced inefficiency in relaxation applications.

In practice the difference equations (2.4) are solved in the ADI algorithm form

$$L_x \Delta Q^* = RHS(2.4) \tag{2.7a}$$

$$L_y \Delta Q^n = \Delta Q^* \tag{2.7b}$$

Variants of this algorithm include higher order difference approximations (including pseudo-spectral right hand side-operators (4)), space varing Δt for steady state applications, addition of viscous terms, etc [c.f. 5-9].

b) Diagonalization

Efforts have been made to reduce the inversion work over and above what is obtained with approximate factorization. In particular, Pulliam and Chaussee[7] have used eigenvector-similarity transforms to reduce the block tridiagonal matrices to scalar tridiagonals(see also Chaussee and Pulliam[8], Steger, Pulliam and Chima[9], Coakley [10]). Specifically, the eigenvectors of the A and B Jacobian matrices, denoted as X and Y, are further approximately factored into the left hand side of equation (2.4) as

$$X\left[I + h\delta_x \Lambda_A{}^n + D_x^{(2)}\right]X^{-1}Y\left[I + h\delta_y \Lambda_B{}^n + D_y^{(2)}\right]Y^{-1}\Delta Q^n = -\Delta t\left[\delta_x E^n + \delta_y F^n + D^{(4)}Q^n\right] \tag{2.8}$$

where

$$\Lambda_A = X^{-1}AX \quad \text{and} \quad \Lambda_B = Y^{-1}BY$$

As Figure 1 illustrates, the number of operations in solving a block tridiagonal (that uses only L-U decomposition) increases with the cube of the block size, a formula given by Merriam [11] indicates the work increases as $\frac{14}{3}m^3 + \frac{9}{2}m^2 - \frac{1}{6}m$ where m is the dimension of the block. Specifically our own coded 4×4 block solver requires 370 operations per entry (i.e. grid point) while the 1×1 tridiagonal needed in (2.8) requires only 9 operations per entry (or 36 operations for 4 scalar tridiagonals). Although the eigenvector matrices must be formed and multiplied through (each 4×4 multiply requires 28 operations per entry), the arithmetic operation count of equations (2.8) is considerablly reduced from that of equations(2.4). For block tridiagonal matrices subject to periodicity conditions and for block pentadiagonal matrices the saving is even more significant.

c) Sound Speed and Velocity Splitting

An alternate method for reducing the inversion (i.e. solution) work was proposed and demonstrated in reference [1].This method was also motivated by similarity transforms, but it does not explicitly use them in the algorithm. Instead, it involves splitting the Jacobian matrices A and B into velocity and sound speed (or pressure parts) as

$$A = A_u + A_c \tag{2.9a}$$

$$B = B_v + B_c \tag{2.9b}$$

such that the eigenvalues $\sigma(A)$ and $\sigma(B)$ are

$$\sigma(A) = \sigma(A_u) + \sigma(A_c) \tag{2.10a}$$

$$\sigma(B) = \sigma(B_u) + \sigma(B_c) \tag{2.10b}$$

with

$$\sigma(A_u) = (u, u, u, u), \quad \sigma(A_c) = (0, c, 0, -c) \tag{2.11a}$$

$$\sigma(B_v) = (v, v, v, v), \quad \sigma(B_c) = (0, 0, c, -c) \tag{2.11b}$$

The matrices A_u, A_c, B_v and B_c were found from a natural reduced form of the equations in nonconservative from. Specifically A_c and B_c were split, as

$$A_c = (\gamma-1)\begin{bmatrix} 0 & 0 & 0 & 0 \\ (u^2+v^2)/2 & -u & -v & 1 \\ 0 & 0 & 0 & 0 \\ a_{41}^c & a_{42}^c & -uv & u \end{bmatrix}, \quad B_c = (\gamma-1)\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ (u^2+v^2)/2 & -u & -v & 1 \\ b_{41}^c & -uv & b_{43}^c & v \end{bmatrix} \tag{2.12}$$

where

$$a_{41}^c = [u(u^2+v^2)/2] - \gamma up/[\rho(\gamma-1)^2]$$

$$a_{42}^c = \gamma p/[\rho(\gamma-1)^2] - u^2$$

$$b_{41}^c = [v(u^2+v^2)/2] - \gamma vp/[\rho(\gamma-1)^2]$$

$$b_{43}^c = \gamma p/[\rho(\gamma-1)^2] - v^2$$

while A_u and B_v are

$$A_u = A - A_c \tag{2.13a}$$

$$B_v = B - B_c \tag{2.13b}$$

This splitting produces matrices A_u and B_v that are more complex than A and B. However, Steger [1] noted that Q is an eigenvector of both matrices, i.e.

$$A_u Q = uQ \tag{2.14a}$$

$$B_v Q = vQ \tag{2.14b}$$

which prompted the ad hoc substitution

$$AQ = (uI + A_c)Q \tag{2.15a}$$

$$BQ = (vI + B_c)Q \tag{2.15b}$$

The matrices $uI + A_c$ and $vI + B_c$ have a reduced form that simplifies inversion compared to A and B.

Insertion of equation (2.15) into the equations for local linearization of the Jacobians (2.6) produces

$$E^{n+1} = E^n + (uI + A_c)^n(Q^{n+1} - Q^n) \tag{2.16a}$$

$$F^{n+1} = F^n + (vI + B_c)^n(Q^{n+1} - Q^n) \tag{2.16b}$$

Ultilizing these linearizations in the basic algorithm equation(2.4)

$$L_x L_y \Delta Q = RHS(2.4) \tag{2.17}$$

gives new left hand side operators L_x and L_y that are easier to solve

$$L_x = \left[I + h\delta_x(uI + A_c)^n + D_x^{(2)}\right] \tag{2.18a}$$

$$L_y = \left[I + h\delta_y(vI + B_c)^n + D_y^{(2)}\right] \tag{2.18b}$$

The end result of this splitting is that the new operators L_x and L_y form matrices that no longer require 4×4 block tridiagonal inversions. Leaving out the dissipation terms to illustrate the structure of these operators, we obtain

$$L_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \Delta t\delta_x \begin{bmatrix} u & 0 & 0 & 0 \\ a_{21}^c & u + a_{22}^c & a_{23}^c & a_{24}^c \\ 0 & 0 & u & 0 \\ a_{41}^c & a_{42}^c & a_{43}^c & u + a_{44}^c \end{bmatrix} \tag{2.19a}$$

$$L_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \Delta t\delta_y \begin{bmatrix} v & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ b_{31}^c & b_{32}^c & v + b_{33}^c & b_{34}^c \\ b_{41}^c & b_{42}^c & b_{43}^c & v + b_{44}^c \end{bmatrix} \tag{2.19b}$$

where a^c and b^c are the respective elements of A_c and B_c given by equation (2.12).

For the L_x operator, for example, the first and third rows decouple from the system and can be solved as scalar tridiagonal matrices with their respective right hand sides. Once these rows are solved, the elements of the first and third columns can be moved to the right hand side. The second and fourth equation remain coupled and are solved as a 2×2 block tridiagonal matrix. The dissipation terms also form diagonal tridiagonals and so do not alter the structure. The block decoupling of the L_y operator is even more conspicuous and is inverted (i.e. solved for) in a similar manner.

Substitution of the left hand side operators given by (2.18) in place of those of (2.7) results in a substantial reduction in arithmetic operations. Our typical 2×2 block tridiagonal requires 55 operations per point, so the overall inversion, including the two scalar tridiagonals, requires 73 operations per entry. Because the two scalar tridiagonals have identical coefficients this work can be even further cut by solving them together.

The matrix splitting (2.15) produces the flux vectors

$$E = AQ = uIQ + A_cQ = E_u + E_c \tag{2.20a}$$

$$F = BQ = vIQ + B_cQ = F_v + F_c \tag{2.20b}$$

where

$$E_u = \begin{bmatrix} \rho u \\ \rho u^2 \\ \rho vu \\ ue \end{bmatrix}, \quad E_c = \begin{bmatrix} 0 \\ p \\ 0 \\ up \end{bmatrix}, \quad F_v = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ ve \end{bmatrix}, \quad F_c = \begin{bmatrix} 0 \\ 0 \\ p \\ vp \end{bmatrix} \tag{2.21}$$

Note that we do not reproduce the matrix splitting (2.9) (with A_c and B_c defined from (2.12)) by taking the Jacobians of (2.21). Surprisingly, linear stability analysis (see Appendix B) as well as numerical experiment have shown that the use of the Jacobian matrices for A_c and B_c is unsatisfactory. However, the same stability analysis and numerical experimentation have confirmed that no numerical stability degradation occurs by using the substitute linearization matrices (2.15). But, the linearization (2.16) is only first order accurate because of the substitution (2.14).

The matrix splitting concept can be extended to three dimensions. In this case the difference equations can be represented by

$$L_xL_yL_x(Q^{n+1} - Q^n) = -\Delta t[\delta_xE^n + \delta_yF^n + \delta_zG^n + D^{(4)}Q^n] \tag{2.21}$$

Associated with each operator are 5×5 block tridiagonal matrices, with each matrix inversion requiring about 700 operations per grid point. Typically the inversion represents 70 to 80 per cent of the overall work per point. Replacing each Jacobian A by $uI + A_c$ etc. reduces the inversion cost from 700 to 82 operations per entry of each block tridiagonal. By combining the scalar tridiagonals, this can be cut to 74 operations.

III. Sound Speed and Velocity Splitting in Steady Generalized Coordinates

The two - dimensional Euler equations can be transformed from Cartesian coordinates to steady curvilinear coordinates using new independent variables

$$\begin{aligned} \tau &= t \\ \xi &= \xi(x,y) \\ \eta &= \eta(x,y). \end{aligned} \tag{3.1}$$

The Euler equations written in steady generalized curvilinear coordinates are

$$\partial_\tau \hat{Q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} = 0 \tag{3.2}$$

$$\hat{Q} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \hat{E} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ U(e+p) \end{bmatrix}, \quad \hat{F} = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ V(e+p) \end{bmatrix}$$

where J is the Jacobian $\xi_x\eta_y - \xi_y\eta_x$ and

$$U = \xi_x u + \xi_y v, \quad V = \eta_x u + \eta_y v$$

are unscaled contravariant velocity components.

Application of the AF implicit scheme (2.4) to equation (3.2) is given by

$$\begin{aligned} \left[I + h\delta_\xi \hat{A}^n + D_\xi^{(2)}\right]\left[I + h\delta_\eta \hat{B}^n + D_\eta^{(2)}\right]\Delta\hat{Q}^n = \\ -\Delta t\left[\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n + D^{(4)}\hat{Q}^n\right] \end{aligned} \tag{3.3a}$$

where

$$D^{(4)} = \epsilon_e \Delta t J^{-1}[(\nabla_\xi \Delta_\xi)^2 + (\nabla_\eta \Delta_\eta)^2]J \tag{3.3b}$$

and

$$D_\xi^{(2)} = -\epsilon_i \Delta t J^{-1}\nabla_\xi \Delta_\xi J, \quad D_\eta^{(2)} = -\epsilon_i \Delta t J^{-1}\nabla_\eta \Delta_\eta J \tag{3.3c}$$

The Jacobian matrices of $\hat{E}$ and $\hat{F}$ can be given in terms of the Jacobian matrices A and B as

$$\hat{A} = \xi_x A + \xi_y B \tag{3.4a}$$

$$\hat{B} = \eta_x A + \eta_y B \tag{3.4b}$$

Making the substitutions of (2.15) into $\hat{A}\hat{Q}$ and $\hat{B}\hat{Q}$ results in

$$\hat{A}\hat{Q} = (UI + \xi_x A_c + \xi_y B_c)\hat{Q} \tag{3.5a}$$

$$\hat{B}\hat{Q} = (VI + \eta_x A_c + \eta_y B_c)\hat{Q} \tag{3.5b}$$

Because of the linear combination of both A_c and B_c being present in Eq.(3.5), only 3×3 reducibility is achieved. When applied in the implicit AF algorithm, this results in a savings as indicated in Ref.1, but not as great a saving as what was achieved in Cartesian coordinates. Moreover, in three dimensions one can only reduce to a 4×4 block tridiagonal.

The transformed equations given above have only been transformed in their independent variables. That is, Cartesian velocity or momentum components have been kept, and, as a result, the so called strong conservation law form is maintained. Suppose, for example, we had used orthogonal body coordinates s and n, and that we transformed the momentum components as well. We would then obtain equations with coordinate source terms, but otherwise the equations would look like their Cartesian counterparts and the Jacobian matrices could be reduced as before. We can achieve this same effect with equation (3.2) by multiplying through by the matrix

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \xi_x & \xi_y & 0 \\ 0 & \eta_x & \eta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

This matrix transforms u,v momentum components to U,V components. With multiplication of (3.2) by C we obtain

$$\partial_\tau J^{-1}\begin{bmatrix} \rho \\ \rho U \\ \rho V \\ e \end{bmatrix} + \partial_\xi J^{-1}\begin{bmatrix} \rho U \\ \rho UU + p(\nabla\xi\cdot\nabla\xi) \\ \rho VU + p(\nabla\xi\cdot\nabla\eta) \\ U(e+p) \end{bmatrix} + \partial_\eta J^{-1}\begin{bmatrix} \rho V \\ \rho UV + p(\nabla\xi\cdot\nabla\eta) \\ \rho VV + p(\nabla\eta\cdot\nabla\eta) \\ V(e+p) \end{bmatrix} = H \tag{3.7}$$

with

$$H = -C[(C_\xi)^{-1}\hat{E} + (C_\eta)^{-1}\hat{F}]$$

These equations look much like their Cartesian counterparts except for the coordinate source term and the appearance of two extra pressure terms.

However, for orthogonal coordinates, $\nabla\xi \cdot \nabla\eta = 0$, and the extra pressure terms then drop out. Although we will not give the details here, one can anticipate from the form of the equations that the flux Jacobians in the implicit algorithm are 2×2 reduciable (here the Jacobian matrices are formed with respect to ρU and ρV components, not ρu and ρv components). So if the coordinate generated source term is either treated explicitly or properly distributed in the L_ξ or L_η operators, the inversion efficiency of the previous section can be obtained.

Generally the transformed coordinates will not be orthogonal and multiplication through by C will not lead to the 2×2 reduced matrix operators. However, if we use a transform matrix that brings out a mixture of contravariant and covariant velocity components, the extra pressure terms will also disappear. Although it may not be immediately obvious, we can again obtain the reduced matrix operator structure without requiring the coordinates to be orthogonal.

Consider the transform matrix $\tilde{C}$ given by

$$\tilde{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \eta_y \ell_2^{-1} & -\eta_x \ell_2^{-1} & 0 \\ 0 & -\xi_y \ell_1^{-1} & \xi_x \ell_1^{-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

where

$$\ell_1 = \sqrt{\nabla\xi \cdot \nabla\xi}$$

$$\ell_2 = \sqrt{\nabla\eta \cdot \nabla\eta}$$

and the inner 2×2 of $\tilde{C}$ transforms u and v into the covariant velocity components

$$\tilde{U} = \ell_2^{-1}(\eta_y u - \eta_x v), \quad \tilde{V} = \ell_1^{-1}(-\xi_y u + \xi_x v)$$

Since the determinant of $\tilde{C}$ is $J\ell_2^{-1}\ell_1^{-1}$, its inverse exists for all mappings of interest and is

$$\tilde{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \ell_2 y_\eta & -\ell_1 y_\xi & 0 \\ 0 & -\ell_2 x_\eta & \ell_1 x_\xi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

Multipling equation (3.2) by $\tilde{C}$ we obtain

$$\partial_\tau \tilde{Q} + \partial_\xi \tilde{E} + \partial_\eta \tilde{F} = \tilde{H} \tag{3.10}$$

where

$$\tilde{H} = -\tilde{C}(\tilde{C}_\xi{}^{-1}\hat{E} + \tilde{C}_\eta{}^{-1}\hat{F})$$

and $\tilde{Q} = \tilde{C}\hat{Q}, \tilde{E} = \tilde{C}\hat{E}, \tilde{F} = \tilde{C}\hat{F}$. The flux vectors written out are

$$\tilde{Q} = J^{-1}\begin{bmatrix}\rho\\ \rho\tilde{U}\\ \rho\tilde{V}\\ e\end{bmatrix}, \quad \tilde{E} = J^{-1}\begin{bmatrix}\rho U\\ \rho\tilde{U}U + Jp/\ell_2\\ \rho\tilde{V}U\\ U(e+p)\end{bmatrix}, \quad \tilde{F} = J^{-1}\begin{bmatrix}\rho V\\ \rho\tilde{U}V\\ \rho\tilde{V}V + Jp/\ell_1\\ V(e+p)\end{bmatrix} \tag{3.11}$$

Unlike equation (3.7) extra pressure terms are not picked up in the momentum equations.

The previous numerical algorithm extended to equation (3.10) is given by

$$\begin{aligned}\left[I + h\delta_\xi \tilde{A}^n + D_\xi^{(2)}\right]\left[I + h\delta_\eta \tilde{B}^n + D_\eta^{(2)}\right]\Delta\tilde{Q}^n = \\ -\Delta t\left[\delta_\xi \tilde{E}^n + \delta_\eta \tilde{F}^n + \tilde{H}^n + D^{(4)}\tilde{Q}^n\right]\end{aligned} \tag{3.12}$$

where we have lagged the source term and where $\tilde{A} = \partial\tilde{E}/\partial\tilde{Q}$ and $\tilde{B} = \partial\tilde{F}/\partial\tilde{Q}$.

It can be verified that $\tilde{A} = \tilde{C}\hat{A}\tilde{C}^{-1}$ and $\tilde{B} = \tilde{C}\hat{B}\tilde{C}^{-1}$. Using these relations as well as (3.5) we find

$$\tilde{A}\tilde{Q} = (UI + \tilde{A}_c)\tilde{Q} \quad \tilde{B}\tilde{Q} = (VI + \tilde{B}_c)\tilde{Q} \tag{3.13}$$

with

$$\tilde{A}_c = (\gamma - 1)\begin{bmatrix}0 & 0 & 0 & 0\\ J(u^2+v^2)/(2\ell_2) & -U & -V\ell_1/\ell_2 & J/\ell_2\\ 0 & 0 & 0 & 0\\ a_{41}^c & a_{42}^c & a_{43}^c & U\end{bmatrix} \tag{3.14a}$$

$$\tilde{B}_c = (\gamma - 1)\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ J(u^2+v^2)/(2\ell_1) & -U\ell_2/\ell_1 & -V & J/\ell_1\\ b_{41}^c & b_{42}^c & b_{43}^c & V\end{bmatrix} \tag{3.14b}$$

where

$$a^c_{41} = U[(u^2+v^2)/2 - \gamma p/(\rho(\gamma-1)^2)]$$
$$a^c_{42} = \ell_2[\gamma p \ell_1^2/(\rho(\gamma-1)^2) - U^2]/J$$
$$a^c_{43} = \ell_1[\gamma p \ell_{12}^2/(\rho(\gamma-1)^2) - UV]/J$$
$$b^c_{41} = V[(u^2+v^2)/2 - \gamma p/(\rho(\gamma-1)^2)]$$
$$b^c_{42} = \ell_2[\gamma p \ell_{12}^2/(\rho(\gamma-1)^2) - UV]/J$$
$$b^c_{43} = \ell_1[\gamma p \ell_2^2/(\rho(\gamma-1)^2) - V^2]/J$$

and

$$\ell_{12} = \sqrt{\nabla\xi \cdot \nabla\eta}$$

Making these substitutions into equation (3.12) we obtain the reduced form of the algorithm

$$\left[I + h\delta_\xi(UI + \tilde{A}_c)^n + D^{(2)}_\xi\right]\left[I + h\delta_\eta(VI + \tilde{B}_c)^n + D^{(2)}_\eta\right]\Delta\tilde{Q}^n = -\Delta t\left[\delta_\xi \tilde{E}^n + \delta_\eta \tilde{F}^n + \tilde{H}^n + D^{(4)}\tilde{Q}\right] \tag{3.15}$$

Finally, to avoid the source term as well as to take advantage of existing codes, one can rewrite these equations in the form

$$\tilde{C}^{-1}\left[I + h\delta_\xi(UI + \tilde{A}_c)^n + D^{(2)}_\xi\right]\left[I + h\delta_\eta(VI + \tilde{B}_c)^n + D^{(2)}_\eta\right]\tilde{C}\Delta\hat{Q}^n = -\Delta t\left[\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n + D^{(4)}\hat{Q}\right] \tag{3.16}$$

The right hand side of this equation is the same as equation (3.3a), while the left hand side maintains the update for $\hat{Q}$ quantities rather than $\tilde{Q}$. The advantage of this last form is that one can readily take an existing code for equation (3.2) and modify only the left hand side operators so as to take advantage of the reduced inversion work. The steady state solution of equation (3.16) is clearly identical to that of equation (3.3a).

IV. Results

The algorithm given by equation (3.16) was tested on a NACA 0012 airfoil under transonic flow conditions in order to verify that no adverse stability effects are incurred by using the matrix reductions. Versions of the basic NASA Ames Research Center AIR2D/ARC2D, which solve equation (3.3a), were modified to the form of equation (3.16). As noted earlier, only that portion of the code which deals with the left hand side operator has to be altered in order to implement the new algorithm. The basic code is described in Refs. [5,6].

A "C-type" topology was used for the airfoil calculations. The grids were generated with either a hyperbolic grid solver [12,13] that imposes orthogonality up to numerical errors of truncation and smoothing, or an algebraic construction that uses a method similar to the control function approach of Eiseman[14]. In all cases, the far field boundaries were placed approximately 20 chords from the body. Boundary conditions on the body, wake, and freestream are further described in Refs.[5,6].

As a first calculation, a relatively coarse grid was used which was generated with the hyperbolic grid solver. This grid has 157 points in the ξ-direction and 33 points in the η-direction (see Figures 2a-2b). The pressure distribution on the NACA 0012 for $M_\infty = .8$ and $\alpha = 0.$ is shown in Figure 2c. This distribution was computed using both the standard algorithm (3.3a) and the reduced matrix form (3.16). Both algorithms were run using Euler implicit differencing which is first order accurate in time. As shown in figure 2d, the residual histories are virtually identical. However, the standard algorithm required 120 CPU seconds of CRAY-XMP time per 1000 iterations while the new algorithm required 54 CPU seconds per 1000 iterations.

Numerical experimentation, in which Δt was adjusted over a wide range, has confirmed that the new algorithm is as stable as the standard algorithm. As an example of the robustness of the formulation using (3.16), a much finer grid was used to compute the previous case. Figures 3a-3b show a view of the grid with 249 points in the ξ-direction and 50 points in the η-direction. The solution on this grid (Figures 3c-3d) was computed using a spacially varying time step scaled by $\frac{1}{1+\sqrt{J}}$ where J is the metric Jacobian. For this calculation, the drag coefficient was both converged to 5 digits in 600 iterations or 68 seconds of CRAY-XMP time. Figure 3e shows the residual history for this calculation.

Recently, Beam and Bailey have reported in private communication [see also 6] that significantly faster convergence can be obtained by incorporat-

ing the fourth order smoothing terms implicitly into the Beam and Warming algorithm. For the standard algorithm, imposing fourth order dissipation implicitly necessitates 4×4 block pentadiagonal inversions which are relatively expensive. The reduced matrix algorithm, however, requires only scalar and 2×2 block pentadiagonal inversions which are much less costly. In fact, a special 2×2 block pentadiagonal inversion algorithm, coded for this algorithm, requires only 107 arithmetic operations per entry.

In testing the sound speed-velocity splitting algorithm with implicit fourth order dissipation, a more advanced form of numerical filter was implemented. This filter uses both second and fourth order differences such that the second order difference is dominant near shock waves. The spectral radius of the flux Jacobian matrices is used as a scaling to the smoothing coefficients in an attempt to mimick the dissipative nature of second order flux split upwind schemes [15]. Full details are given in reference [6].

As a test case, an algebraic grid was used to compute the flow field around a NACA 0012 airfoil at $M_\infty = .8$ and $\alpha = 1.25$ degrees. The grid has 193 points in the ξ-direction and 33 points in the η-direction. The grid and flow field solution are shown in figures 4a-4d. The residual history (figure 4e) shows the rapid convergence. For this particular case, the CPU time on the CRAY-XMP was 93 seconds per 1000 iterations. This compares with 65 seconds per 1000 iterations using the tridiagonal version of the new algorithm with constant smoothing coefficients which had much slower convergence. It should be noted that the smoothing filter used for the pentadiagonal version required 16 seconds more CPU time than the same smoothing with a constant coefficient filter.

Our computational results verified that the reduced algorithm was every bit as numerically stable as the standard algorithm. This experience is similar to what was observed previously by Steger on a algorithm version that reduced to a 3×3 block and a scalar. Here, with reduction to the 2×2 block, just over a factor of two was saved in overall computer time by using the reduced algorithm (3.16) in place of (3.3a). The precise savings that can be obtained is, of course, machine and coding dependent and a larger improvement should be obtained in three dimensions.

V. Conclusions

By substituting similar reducible matrices for the Jacobian matrices of the flux vectors, a substantial reduction has been achieved in the arithmetic operations needed in solving approximate factorization implicit al-

gorithms such as Beam-Warming. Numerical calculations indicate that the new similarity split matrices can apparently be used without any loss of numerical stability, although time accuracy can be degraded unless additional steps are taken. The new method can be readily retrofit within existing codes, and can likely be extended to viscous flow calculations as well.

References

(1)Steger, J. L. Coefficient Matrices for the Implicit Finite Difference Solution of the Inviscid Fluid Conservation Law Equations, Computer Methods in Applied Mechanics and Engineering, vol. 13, (1978) pp.175-188.

(2) Beam, R. M. and Warming, R. F. An Implicit Finite-Difference Algorith for Hyperbolic Systems in Conservation-Law-Form, J. Comp. Phys., Vol. 22, Sept. 1976, pp.87-110.

(3) Warming, R. F. and Beam, R. M. On the Construction and Application of Implicit Factored Schemes for Conservation Laws, Symposium of CFD, New York, April 16-17, 1977, SIAM-AMS Proceedings, vol. 11, 1977.

(4) Reddy, K. C. Pseudospectral Approximation in a Three-Dimension Navier-Stokes Code. AIAA J., Vol. 21, No.8. August,1983.

(5) Steger, J. L. Implicit Finite Difference Simulation of Flow about Arbitrary Two Dimensional Geometries. AIAA J., vol. 16, No.7, July 1978.

(6) Pulliam, T. H., Euler and Thin Layer Navier Stokes Codes : ARC2D. ARC3D, Notes for COMPUTATIONAL FLUID DYNAMICS USER'S WORKSHOP, The University of Tennessee Space Institute, Tullahoma, Tennessee, March 12-16, 1984.

(7) Pulliam, T. H. and Chaussee, D. S., A Diagonal Form of an Implicit Approximate-Factorization Algorithm, J. Comp. Phys., Vol. 39, No.2, Feb.1981, p.347.

(8) Chaussee, D. S. and Pulliam, T. H., A Diagonal Form of an Implicit Approximate Factorization Algorithm with Application to a Two Dimensional Inlet, AIAA J. Vol. 19, No.2, FEb. 1981, p.153.

(9) Steger, J. L., Pulliam, T. H. and Chima, R. V., An Implicit Finite Difference Code for Inviscid and Viscous Cascade Flow, AIAA paper 80-1427, 1980.

(10)Coakley, T. Numerical Method for Gas Dynamics Combining Characteristics and Conservation Concepts, AIAA Paper 81-1257, 1981.

(11)Merriam, M. L., On the Inversion of Block-Tridiagonals without Storage Constraints, NASA TM-84228, March 1982.

(12)Steger, J. L. and Chaussee, D., Generation of Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations, SIAM J. Sci. Stat. Comput., Vol 1, December 1980, pp 431-437.

(13)Kinsey, D. and Barth, T. J., Description of a Hyperbolic Grid Generating Procedure for Arbitrary Two-Dimensional Bodies, AFWAL-TM, to appear.

(14)Eiseman, P., Geometric Methods in Computational Fluid Dynamics, ICASE Report 80-11, April 18,1980.

(15)Steger, J. L., A Preliminary Study of Relaxation Methods for the Inviscid Conservative Gasdynamics Equations Using Flux Vector Splitting, NASA CR-3415, March 1981.

Appendix A. Extension of Sound Speed and Velocity Splitting to the Navier Stokes Equations

The sound speed and velocity splitting concept can be applied to the Navier Stokes equations because of the special form of the viscous Jacobians. The Navier Stokes equations in strong conservation law form can be represented as

$$\partial_t Q + \partial_x E + \partial_y F = \partial_x E_{vx} + \partial_x E_{vy} + \partial_y F_{vx} + \partial_y F_{vy} \tag{A.1}$$

where Q, E, and F have their previous definitions, the viscous flux terms are given by

$$E_{vx} = \begin{bmatrix} 0 \\ (4/3)\mu u_x \\ \mu v_x \\ E_{vx4} \end{bmatrix}, \quad E_{vy} = \begin{bmatrix} 0 \\ -(2/3)\mu v_y \\ \mu u_y \\ E_{vy4} \end{bmatrix} \tag{A.2a}$$

$$F_{vx} = \begin{bmatrix} 0 \\ \mu v_x \\ -(2/3)\mu u_x \\ F_{vx4} \end{bmatrix}, \quad F_{vy} = \begin{bmatrix} 0 \\ \mu u_y \\ (4/3)\mu v_y \\ F_{vy4} \end{bmatrix} \tag{A.2a}$$

with

$$E_{vx4} = (\mu/2)\partial_x[(4/3)u^2 + v^2] + \mu\beta\partial_x c^2$$

$$E_{vy4} = \mu v u_y - (2/3)\mu u v_y$$

$$F_{vx4} = \mu u v_x - (2/3)\mu v u_x$$

$$F_{vy4} = (\mu/2)\partial_y[u^2 + (4/3)v^2] + \mu\beta\partial_y c^2$$

and

$$\beta = Pr^{-1}(\gamma - 1)^{-1}$$

Beam and Warming have developed a general class of integration techniques for equation (A.1) which is illustrated here for Euler implicit time differencing as

$$\begin{aligned} &\Delta Q^n - \Delta t[\delta_x \Delta E^n + \delta_y \Delta F^n - \bar{\delta}_x \Delta E^n_{vx} - \bar{\delta}_y \Delta F^n_{vy}] \\ &\quad = -\Delta t[\delta_x E^n + \delta_y F^n - \bar{\delta}_x (E^n_{vx} + E^n_{vy}) - \bar{\delta}_y (F^n_{vx} + F^n_{vy})] \end{aligned} \tag{A.3}$$

where $\Delta Q^n = Q^{n+1} - Q^n$ and $\tilde{\delta}$ is a midpoint operator. In this algorithm the cross derivative flux terms E_{vy} and F_{vx} are treated explicitly. As discussed by Beam and Warming, this circumvents the need to include an implicit viscous Jacobain for these cross terms, and it also maintains unconditional stablility for the scalar model equation.

The spacial flux terms have been locally linearized in time. In locally linearizing these terms Beam and Warming expand the viscous terms in a Taylor series using both the function and its derivative, as for example

$$\Delta E^n_{vx} = E^{n+1}_{vx} - E^n_{vx} = \left[\frac{\partial E_{vx}}{\partial Q}\right]^n (Q^{n+1} - Q^n) + \left[\frac{\partial E_{vx}}{\partial Q_x}\right]^n \partial_x (Q^{n+1} - Q^n) \tag{A.4}$$

Alternately, these terms can be expanded in terms of Q alone, for example

$$\Delta E^n_{vx} = \left[\frac{\partial E_{vx}}{\partial Q}\right]^n (Q^{n+1} - Q^n) \tag{A.5}$$

where $\frac{\partial E_{vx}}{\partial Q}$ is now a differential operator. This latter from will be used here. Typically μ is not linearized in time and the Prandtl number is treated as if it where a constant value as far as time linerization is concerned. For a term such as $\mu \partial_y u$, μ is frozen in time and $\mu_0 \partial_y u$ expanded such as

$$\mu_0 \partial_y (\rho u/\rho) = \mu_0 \partial_y [(\rho_0 u_0/\rho_0) - (\rho_0 u_0/{\rho_0}^2)(\rho - \rho_0) + {\rho_0}^{-1}(\rho u - \rho_0 u_0)]$$

or in general

$$\partial_x E_{vx} = \partial_x [E_{vx}|_0 + \mu_0 \partial_x (A_{vx}|_0 (Q - Q_0))]_0 \tag{A.6a}$$

$$\partial_y F_{vy} = \partial_y [F_{vy}|_0 + \mu_0 \partial_y (B_{vy}|_0 (Q - Q_0))]_0 \tag{A.6b}$$

with

$$A_{vx} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -(4/3)(u/\rho) & (4/3)\rho^{-1} & 0 & 0 \\ -(v/\rho) & 0 & \rho^{-1} & 0 \\ a_{x1} & a_{x2} & a_{x3} & \beta/\rho \end{bmatrix}, \quad B_{vy} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -(u/\rho) & \rho^{-1} & 0 & 0 \\ -(4/3)(v/\rho) & 0 & (4/3)\rho^{-1} & 0 \\ b_{x1} & b_{x2} & b_{x3} & \beta/\rho \end{bmatrix} \tag{A.7}$$

and

$$a_{x1} = -(4/3)[u^2/\rho] - [v^2/\rho] + \beta[-(e/\rho) + (u^2 + v^2)/\rho]$$

$$a_{x2} = [(4/3) - \beta](u/\rho)$$

$$a_{x3} = (1 - \beta)(v/\rho)$$

$$b_{y1} = -[u^2/\rho] - (4/3)[v^2/\rho] + \beta[-(e/\rho) + (u^2 + v^2)/\rho]$$

$$b_{y2} = (1 - \beta)(u/\rho)$$

$$b_{y3} = [(4/3) - \beta](v/\rho)$$

With introduction of the sound speed and velocity matrix substitutions for the convection Jacobian matrices, we can obtain operators L_x and L_y that are as reducible as before because of the form of A_{vx} and B_{vy}. To illustrate this we will drop the numerical dissipation terms as before. Then Eq.(A.3) becomes

$$L_x L_y \Delta Q^n = -\Delta t\left[\delta_x E^n + \delta_y F^n - \bar{\delta}_x(E^n_{vx} + E^n_{vy}) - \bar{\delta}_y(F^n_{vx} + F^n_{vy})\right] \tag{A.8}$$

where

$$L_x = I + \Delta t \delta_x \left(\begin{bmatrix} u & 0 & 0 & 0 \\ a^c_{21} & u + a^c_{22} & a^c_{23} & a^c_{24} \\ 0 & 0 & u & 0 \\ a^c_{41} & a^c_{42} & a^c_{43} & u + a^c_{44} \end{bmatrix} + \delta_x \begin{bmatrix} 0 & 0 & 0 & 0 \\ a^v_{21} & a^v_{22} & 0 & 0 \\ a^v_{31} & 0 & a^v_{33} & 0 \\ a^v_{41} & a^v_{42} & a^v_{43} & a^v_{44} \end{bmatrix} \right)$$

$$L_y = I + \Delta t \delta_y \left(\begin{bmatrix} v & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ b^c_{31} & b^c_{32} & v + b^c_{33} & b^c_{34} \\ b^c_{41} & b^c_{42} & b^c_{43} & v + b^c_{44} \end{bmatrix} + \delta_y \begin{bmatrix} 0 & 0 & 0 & 0 \\ b^v_{21} & b^v_{22} & 0 & 0 \\ b^v_{31} & 0 & b^v_{33} & 0 \\ b^v_{41} & b^v_{42} & b^v_{43} & b^v_{44} \end{bmatrix} \right)$$

Here I is the identity matrix and a^c etc are elements of A^c etc. Keep in mind that the viscous terms have been second order accurate differenced to maintain tridiagonal structure using midpoint operators $\bar{\delta}$.

We can now examine the structure of L_x for example, and show that the equations uncouple to two scalar and one 2×2 block tridiagonal. The first row of L_x is clearly uncoupled and can be solved for as a scalar tridiagonal. Its solution is then used to update the right hand side (RHS) of rows 2, 3, and 4. With the first column of the matrices brought to the RHS, the third row is seen to be uncoupled and can thus be solved as another scalar tridiagonal. The RHS terms of the second and fourth rows can now be updated and what

remains is a 2×2 block tridiagonal to be solved. The L_y operator inverts in a similar fashion.

Thus, the addition of the viscous terms in Cartesian coordinates does not prevent the decoupling technique that was successfully used for the inviscid gasdynamic equations. Preliminary work shows that, using the local transformation of Section III, the same decoupling may be possible for the Navier Stokes equations in generalized coordinates. This matter is being further investigated.

Appendix B.Stability of Sound Speed and Velocity Splitting

In order to study the stablility of the algorithm proposed in Section II, the one dimensional Euler equations will be tested for linear stability using a Fourier (or matrix) analysis.

The one dimensional Euler equations in Cartesian coordinates are given by

$$\partial_t Q + \partial_x E = 0 \tag{B.1}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ e \end{bmatrix}, \quad E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(e+p) \end{bmatrix}, \tag{B.2}$$

A frozen coefficient form of Eq.(B.1) is given by

$$\partial_t Q + A^* \partial_x E = 0 \tag{B.3}$$

where A is the true Jacobian, $[\partial E/\partial Q]$, and * denotes that A is evaluated using a frozen value of Q, Q^*. Implicit first order accurate differencing of Eq.(B.3) is given in delta form as

$$\left[I + hA^* \delta_x\right](Q^{n+1} - Q^n) = -hA^* \delta_x Q^n \tag{B.4}$$

In our sound speed - velocity splitting scheme we replace the A^* matrix of the left hand side operator with the similarity matrices $u^* I + A_c^*$ to obtain

$$\left[I + h(u^* I + A_c^*)\delta_x\right](Q^{n+1} - Q^n) = -hA^* \delta_x Q^n \tag{B.5}$$

The right hand side matrix A^* is not altered in our procedure, and must not be changed in Eq.(B.5) as it represents the correct linearization of E about Q^*. The question to be addressed is Eq.(B.5) as stable as Eq.(B.4) now that the implicit operator as been altered?

If we were to perform a stability analysis of Eq.(B.4) we would diagonalize A^* using its eigenvectors. We can then readily perform the stability analysis for difference equations corresponding to scalar partial differential equations. For Eq.(B.5) we can use the eigenvectors of A_c^* to diagonalize $u^* I + A_c^*$, but the right hand side matrix will not be brought into diagonal form. Nevertheless, we find that the resulting equations can still be analyzed.

The matrix A_c for one dimensional flow is given by

$$A_c = (\gamma - 1)\begin{bmatrix} 0 & 0 & 0 \\ u^2/2 & -u & 1 \\ a_{41}^c & a_{42}^c & u \end{bmatrix} \tag{B.6}$$

where

$$a_{41}^c = u[u^2/2 - c^2/(\gamma - 1)^2]$$
$$a_{42}^c = -u^2 + c^2/(\gamma - 1)^2$$

Eigenvectors matrices of A_c are given by

$$X^{-1} = \begin{bmatrix} -[(\gamma - 1)u^2 + 2cu]/(4c) & [(\gamma - 1)u + c]/(2c) & -(\gamma - 1)/(2c) \\ [(\gamma - 1)u^2 - 2cu]/(4c) & -[(\gamma - 1)u - c]/(2c) & (\gamma - 1)/(2c) \\ 1 & 0 & 0 \end{bmatrix} \tag{B.7}$$

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & u \\ u - (c/(\gamma - 1)) & u + (c/(\gamma - 1)) & u^2/2 \end{bmatrix} \tag{B.8}$$

and

$$\Lambda_{A_c} = X^{-1} A_c X = \begin{bmatrix} -c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{B.9}$$

Multiplying Eq.(B.5) by the frozen-coefficient inverse of the eigenvectors $(X^*)^{-1}$ gives

$$\left[I + h(u^* I + \Lambda_{A_c}^*)\delta_x\right](X^*)^{-1}(Q^{n+1} - Q^n) = -h(X^*)^{-1} A^* X^* \delta_x (X^*)^{-1} Q^n \tag{B.10}$$

or in nondelta form with $W = (X^*)^{-1}Q$

$$\left[I + h(u^* I + \Lambda_{A_c}^*)\delta_x\right] W^{n+1} = I - h[(X^*)^{-1} A^* X^* - u^* I - \Lambda_{A_c}^*]\delta_x W^n \tag{B.11}$$

However, $(X^*)^{-1}A^*X^* - u^*I - \Lambda_{A_c}^*$ is a very simple matrix

$$(X^*)^{-1} A^* X^* - u^* I - \Lambda_{A_c}^* = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{B.12}$$

Assuming periodic boundary conditions so as to use a Fourier stability analysis (or the matrix stablity method using circulant matrices), the finite difference operator δ_x can be transformed to

$$\frac{2isin(\theta_j)}{2\Delta x} = \kappa$$

and $\tilde{W}$ denotes Fourier transformation of W (i.e. transformation via the eigenvectors of a circulant matrix).

Applying the Fourier method to Eq.(B.11) and taking the inverse of the left hand side operator gives

$$\tilde{W}^{n+1} = \begin{bmatrix} \frac{1}{1+ih\kappa(u+c)} & 0 & 0 \\ 0 & \frac{1}{1+ih\kappa(u-c)} & 0 \\ 0 & 0 & \frac{1}{1+ih\kappa u} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -ih\kappa & -ih\kappa & 1 \end{bmatrix} \tilde{W}^n \qquad \text{(B.13)}$$

or

$$\tilde{W}^{n+1} = \begin{bmatrix} \frac{1}{1+ih\kappa(u+c)} & 0 & 0 \\ 0 & \frac{1}{1+ih\kappa(u-c)} & 0 \\ \frac{-ih\kappa}{1+ih\kappa u} & \frac{-ih\kappa}{1+ih\kappa u} & \frac{1}{1+ih\kappa u} \end{bmatrix} \tilde{W}^n \qquad \text{(B.14)}$$

The amplification matrix associated with Eq.(B.14) is lower triangular. Therefore, its eigenvalues are the diagonal elements, and these clearly have modulus less than one for any positive value of $h = \Delta t$. In fact, these eigenvalues are identical to those obtained with the standard scheme given by Eq.(B.4). Thus the necessary condition for stability is met for any values of h. A sufficient condition for stability requires bounding the norm of the 3×3 amplification matrix. Observing the 3,1 and 3,2 elements it is clear that the ℓ_∞ norm can be unbounded for $u \to 0$ and $h \to \infty$, however, even in this norm powers of the amplification matrix will be quickly bounded unless u is identically zero.

It is interesting to note that if one were to form A_c using the Jacobian of E_c, the resulting algorithm is unconditionally unstable. In this case A_c is given as

$$A_c = (\gamma - 1)\begin{bmatrix} 0 & 0 & 0 \\ u^2/2 & -u & 1 \\ a^c_{41} & a^c_{42} & u \end{bmatrix} \qquad \text{(B.15)}$$

where

$$a^c_{41} = u[u^2/2 - c^2/\gamma(\gamma - 1)]$$

$$a^c_{42} = -u^2 + c^2/\gamma(\gamma - 1)$$

Proceding in a similar fashion as above, the resultant representation in Fourier space for (B.5) is

$$\tilde{W}^{n+1} = \begin{bmatrix} \frac{2\beta + ih\kappa c}{2\beta(1+ih\kappa(u+\alpha c))} & \frac{ih\kappa c}{2\beta(1+ih\kappa(u+\alpha c))} & 0 \\ \frac{-ih\kappa c}{2\beta(1+ih\kappa(u-\alpha c))} & \frac{2\beta - ih\kappa c}{2\beta(1+ih\kappa(u-\alpha c))} & 0 \\ \frac{-ih\kappa}{1+ih\kappa u} & \frac{-ih\kappa}{1+ih\kappa u} & \frac{1}{1+ih\kappa u} \end{bmatrix} \tilde{W}^n \qquad \text{(B.16)}$$

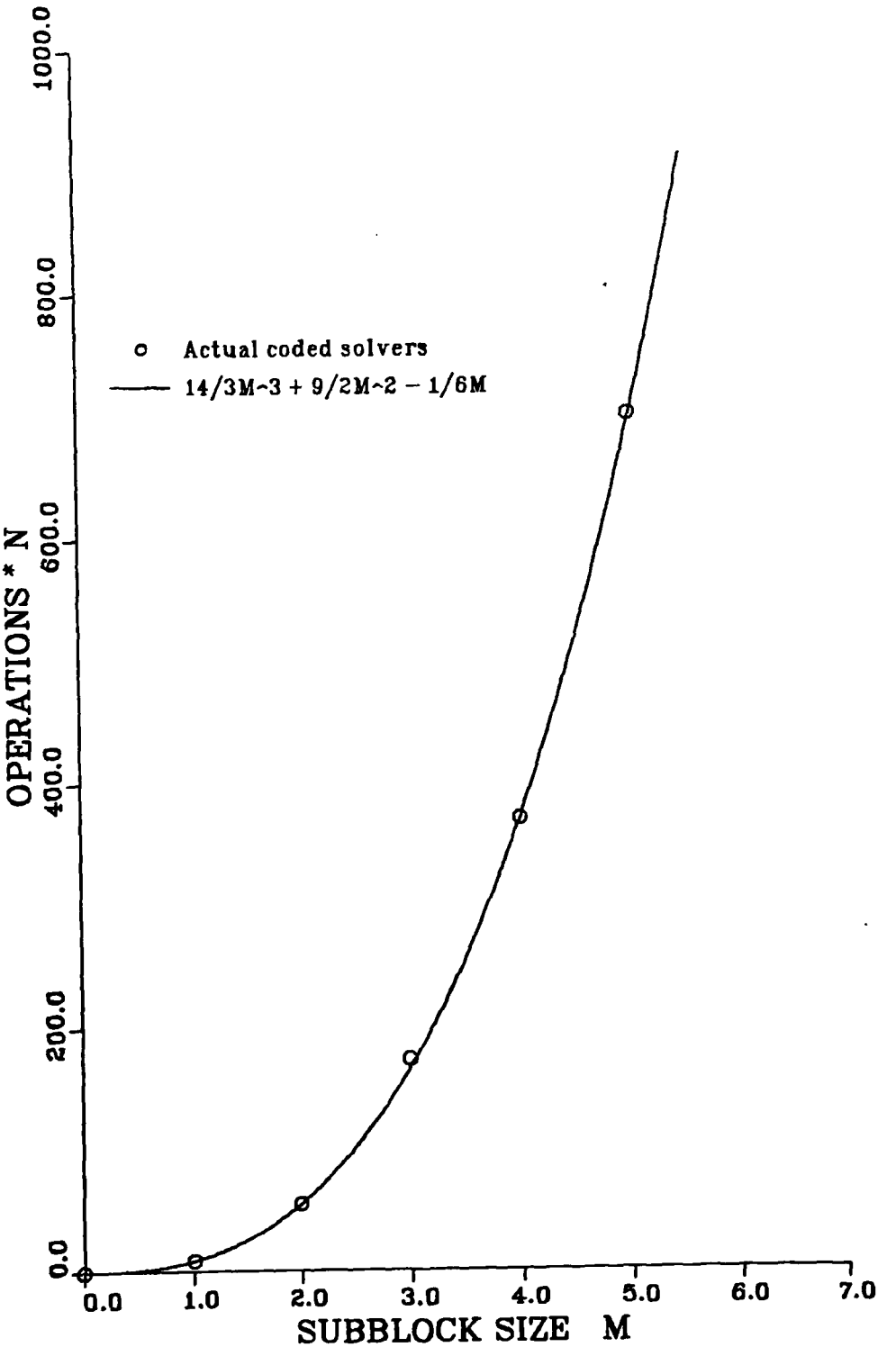
where

$$\alpha = \sqrt{\gamma(\gamma-1)} \quad \beta = \sqrt{\frac{\gamma-1}{\gamma}}$$

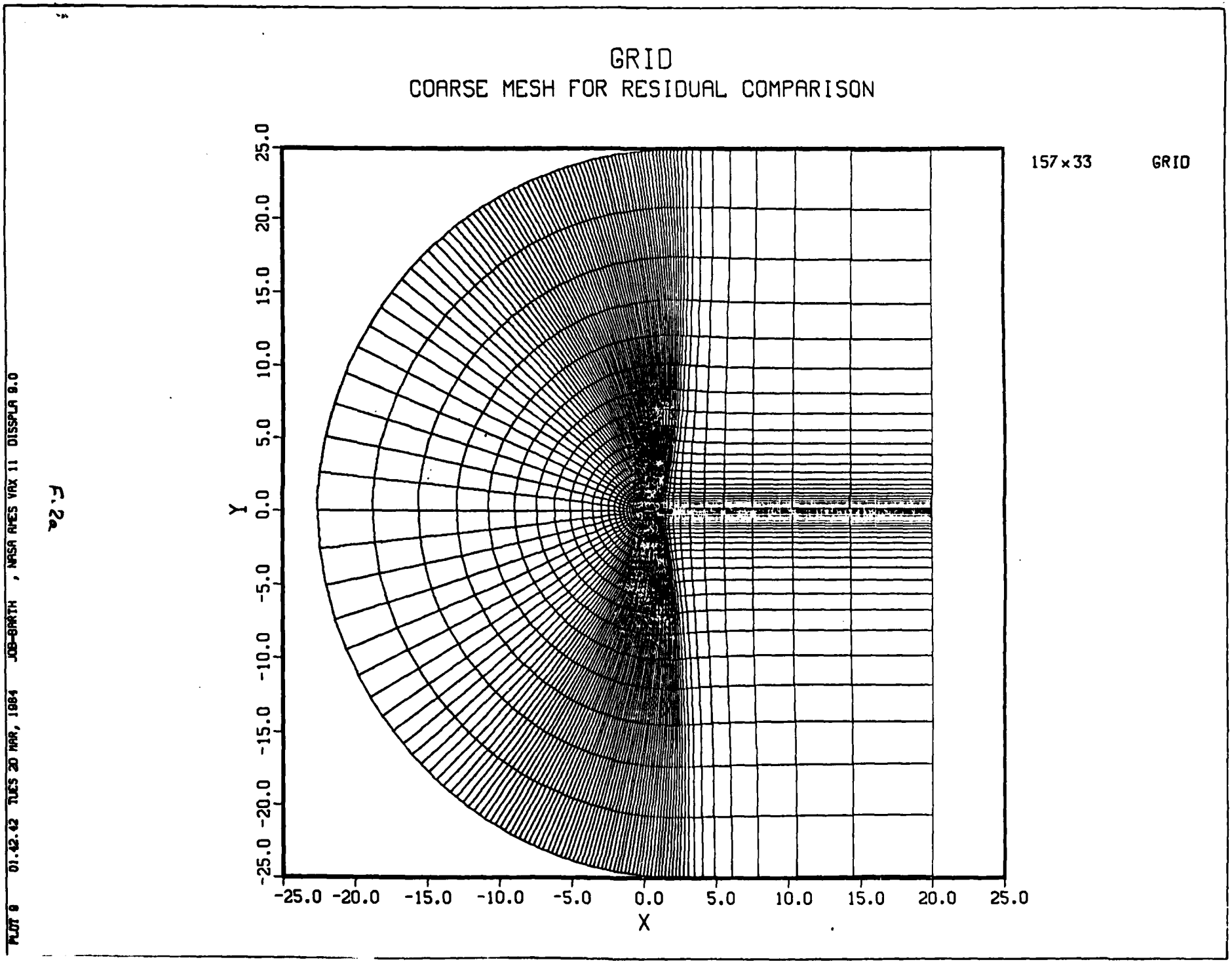
The maximum eigenvalue of the amplification matrix in (B.16) can be shown to be greater than one for all Δt's greater that zero and the necessary condition for stability can not be met. In actual numerical experimentation where numerical dissipation is added, small values of Δt could be used to obtain stable results.

FIGURES

OPERATION COUNTS FOR BLOCK SOLVERS

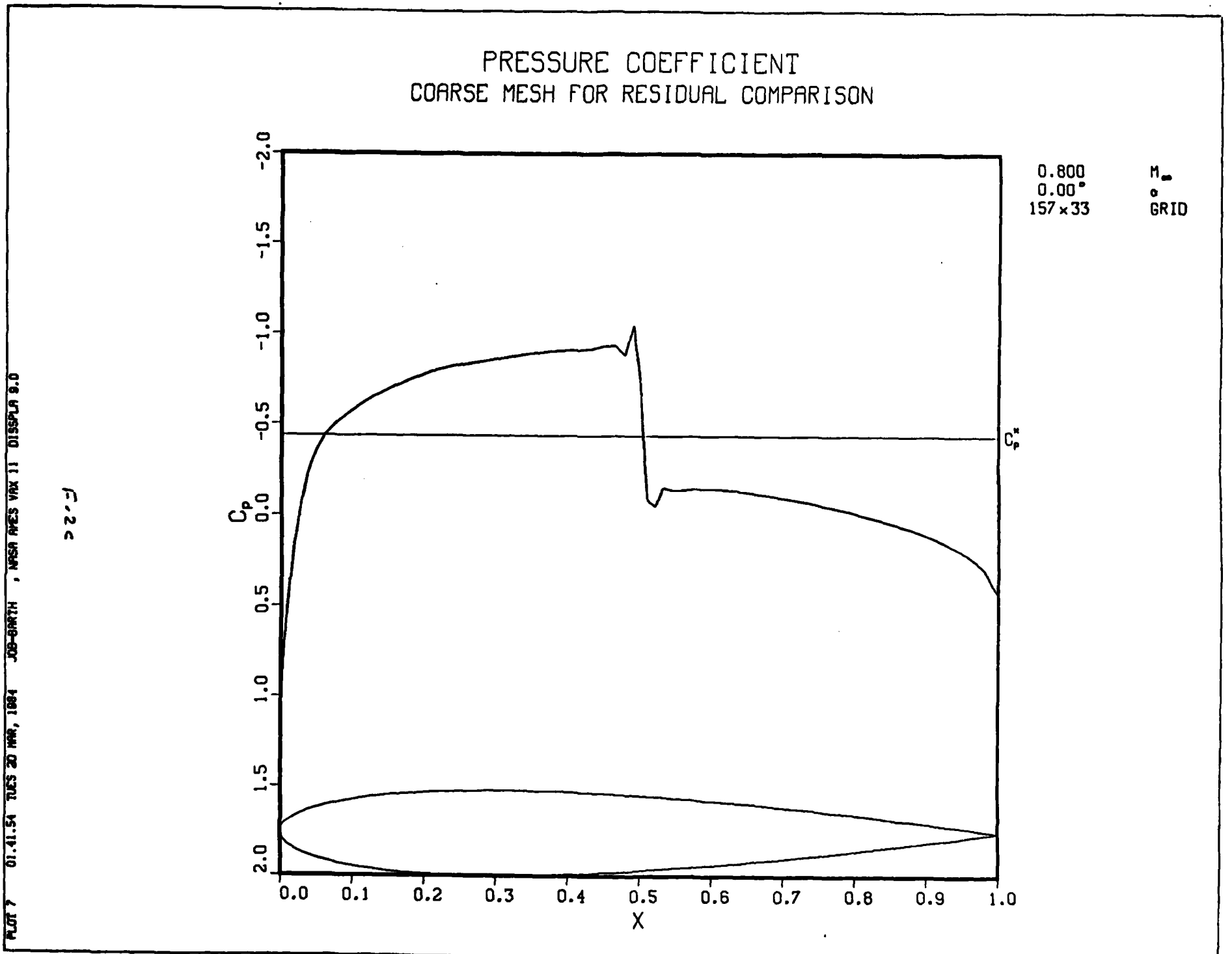


F.1

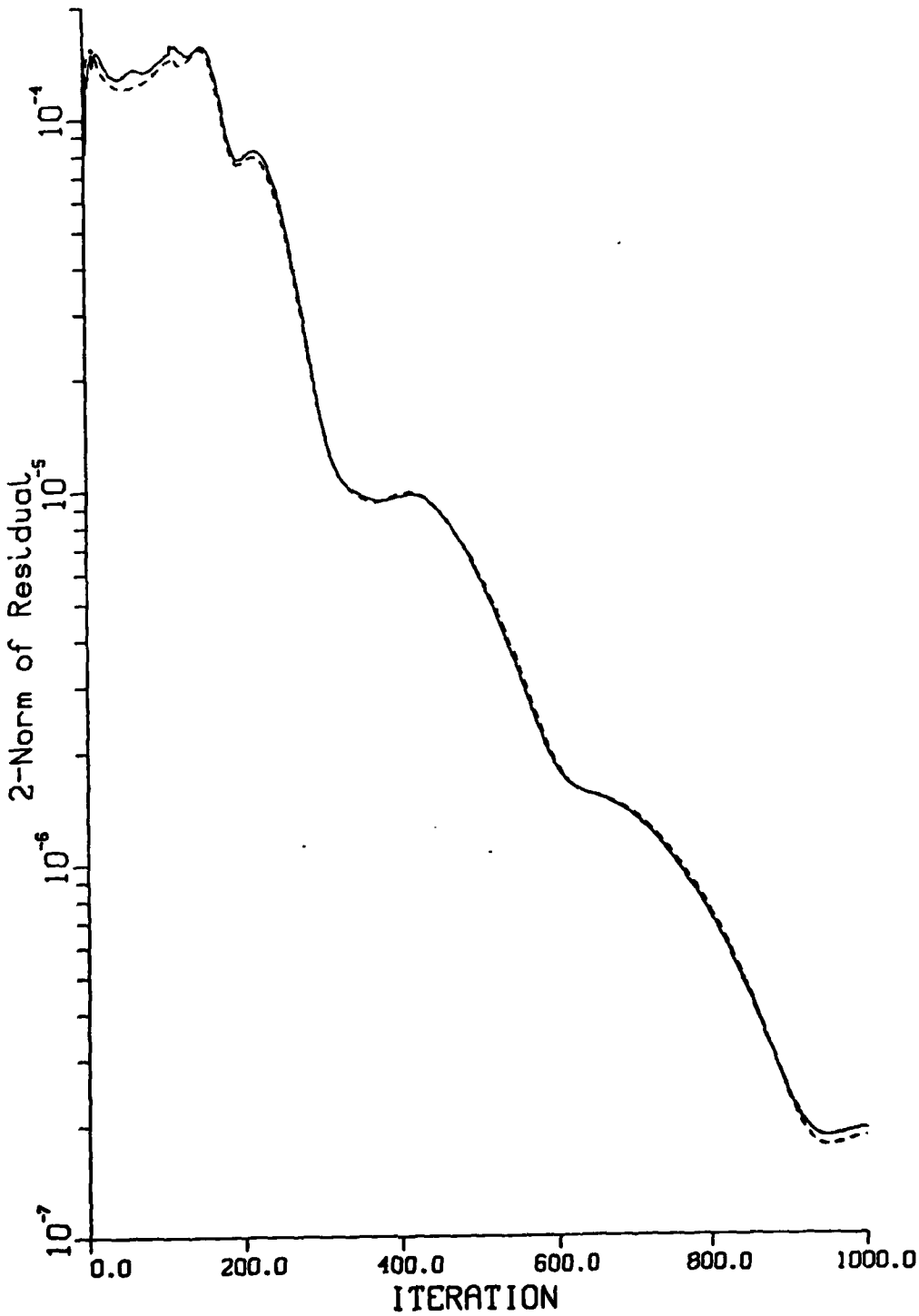

GRID
COARSE MESH FOR RESIDUAL COMPARISON
157 x 33
GRID
Y
X
25.0
20.0
15.0
10.0
5.0
0.0
-5.0
-10.0
-15.0
-20.0
-25.0
-25.0 -20.0 -15.0 -10.0 -5.0 0.0 5.0 10.0 15.0 20.0 25.0
PLOT 0 01:42:42 TUES 20 MAR, 1984 JOB-BARTH , NASA AMES VAX 11 DISSPLA 8.0

F.2a

F. 2b


PRESSURE COEFFICIENT
COARSE MESH FOR RESIDUAL COMPARISON
0.800
0.00°
157 x 33
M∞
α
GRID
C_p
C_{p^*}
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X
PLOT 7 01:41:54 TUES 20 MAR, 1984 JOB-BARTH , NASA AMES VAX 11 DISSPLA 8.0

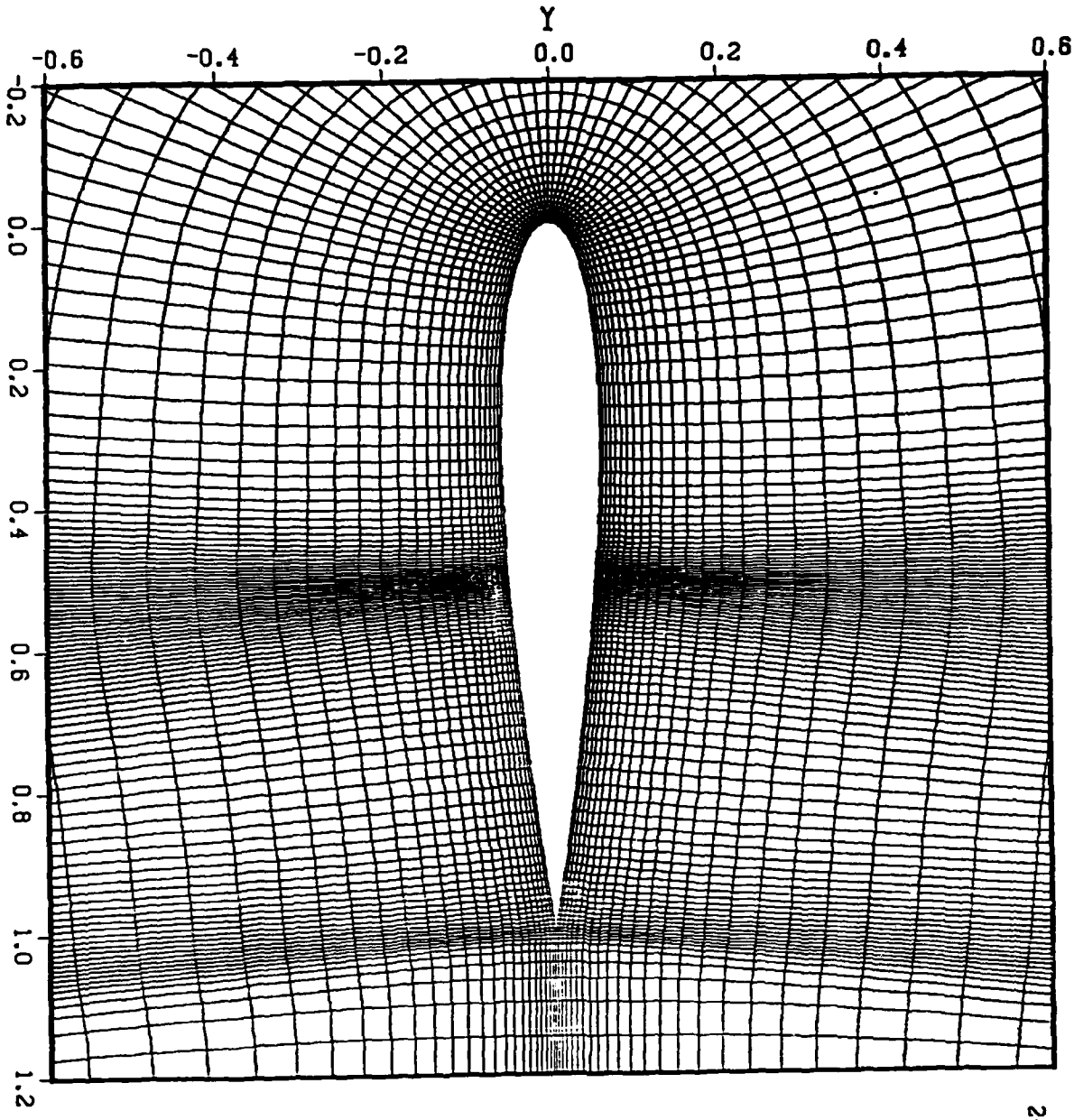
RESIDUAL COMPARISON
NO ALGORITHM ENHANCEMENTS
DT=.2, SMU=1.2,SMUIM=2.5
157X33 C-MESH, M=.80,ALPHA=0.



F.2d

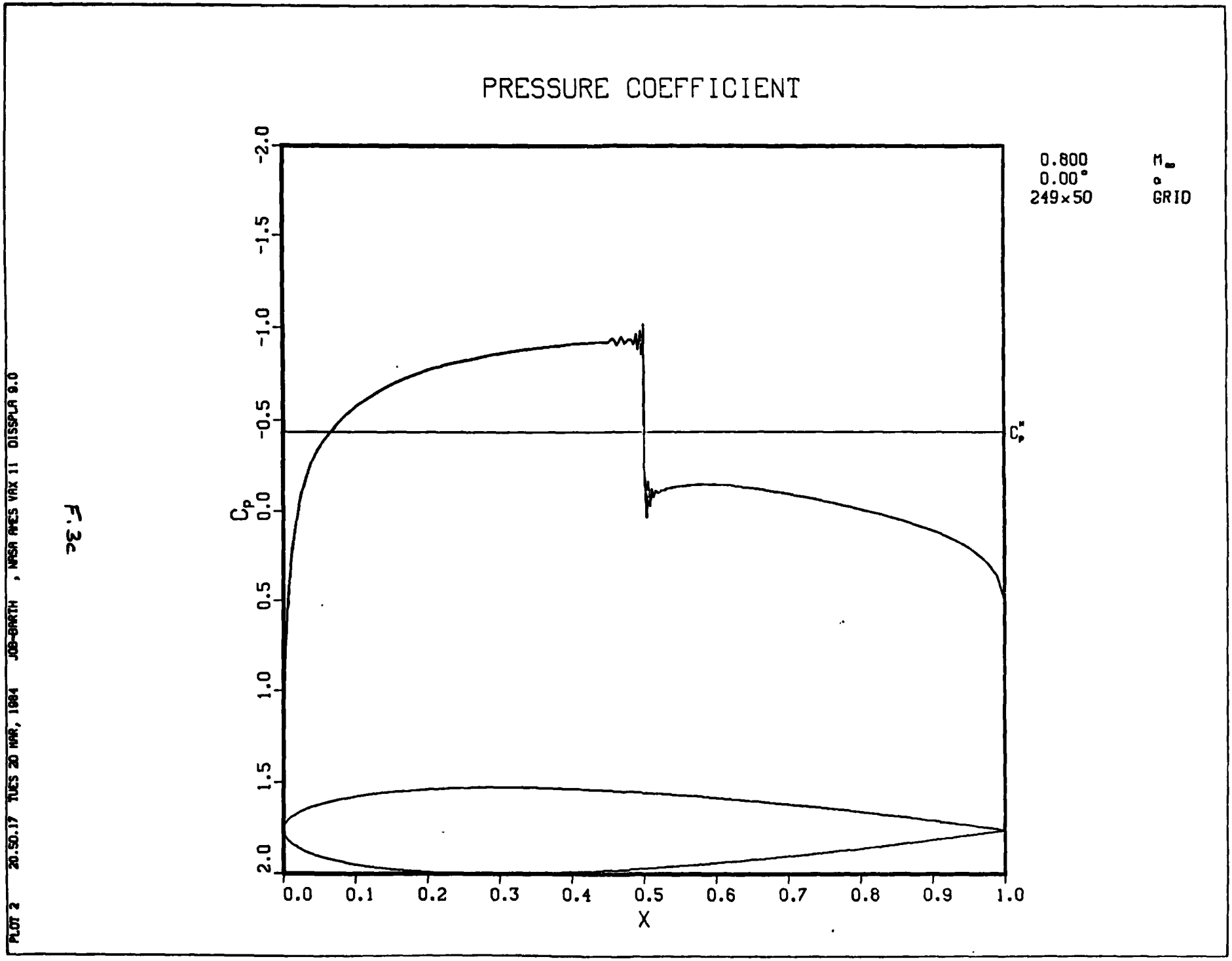
GRID
249×50
GRID
20.0
15.0
10.0
5.0
0.0
-5.0
-10.0
-15.0
-20.0
X
Y
-20.0
-15.0
-10.0
-5.0
0.0
5.0
10.0
15.0
20.0
PLOT 3 20:39:41 TUES 20 MAR, 1984 JOB-BARTH , NASA AMES VAX 11 DISSPLA 9.0

F. 3a



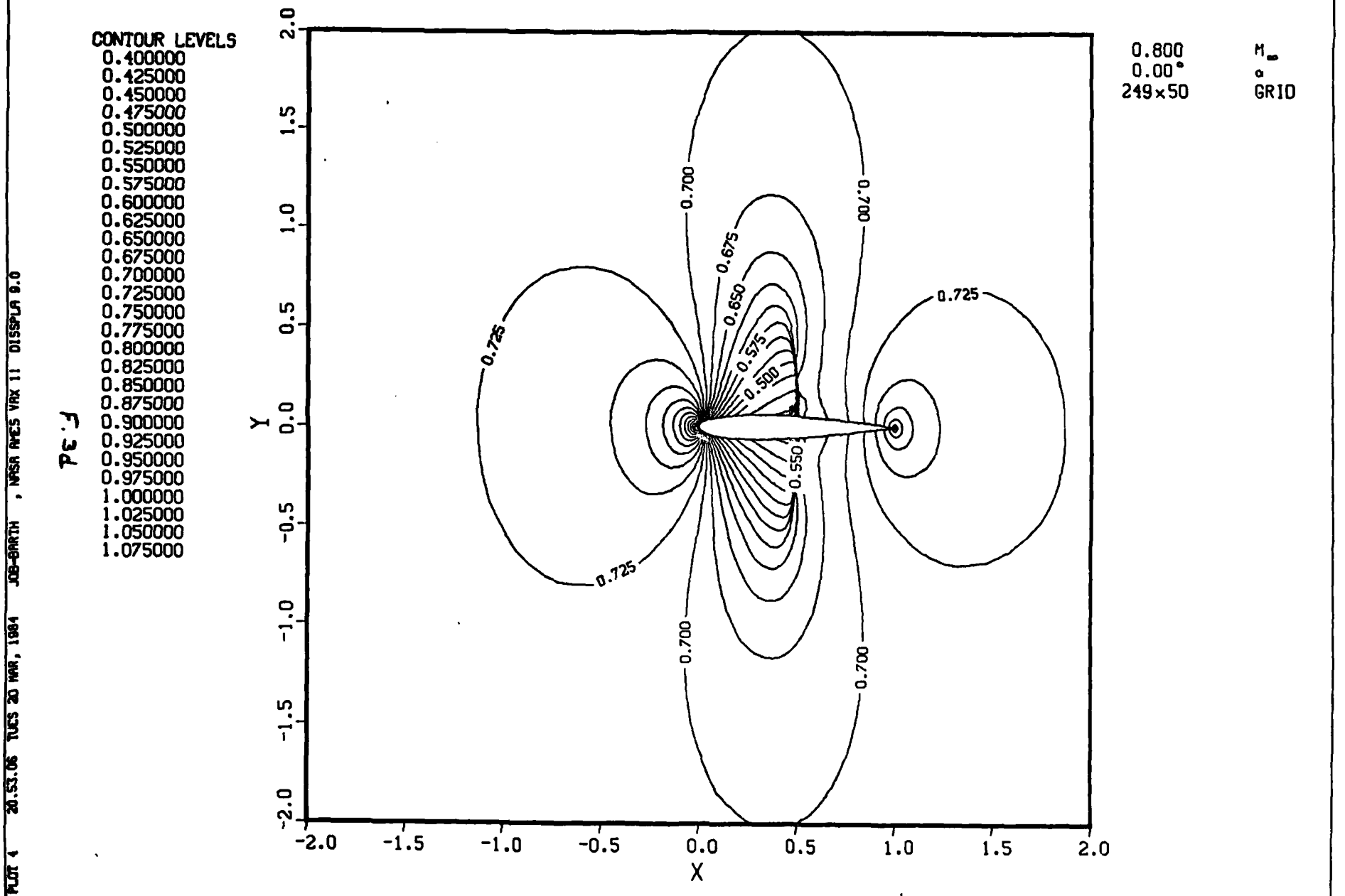
GRID

249×50 GRID


PRESSURE COEFFICIENT
0.800 M_∞
0.00° α
249×50 GRID
C_p
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X
PLOT 2 20.50.17 TUES 20 MAR, 1984 JOB-BARTH , NASA AMES VAX 11 DISSPLA 9.0

F.3c

PRESSURE CONTOURS



F.3d

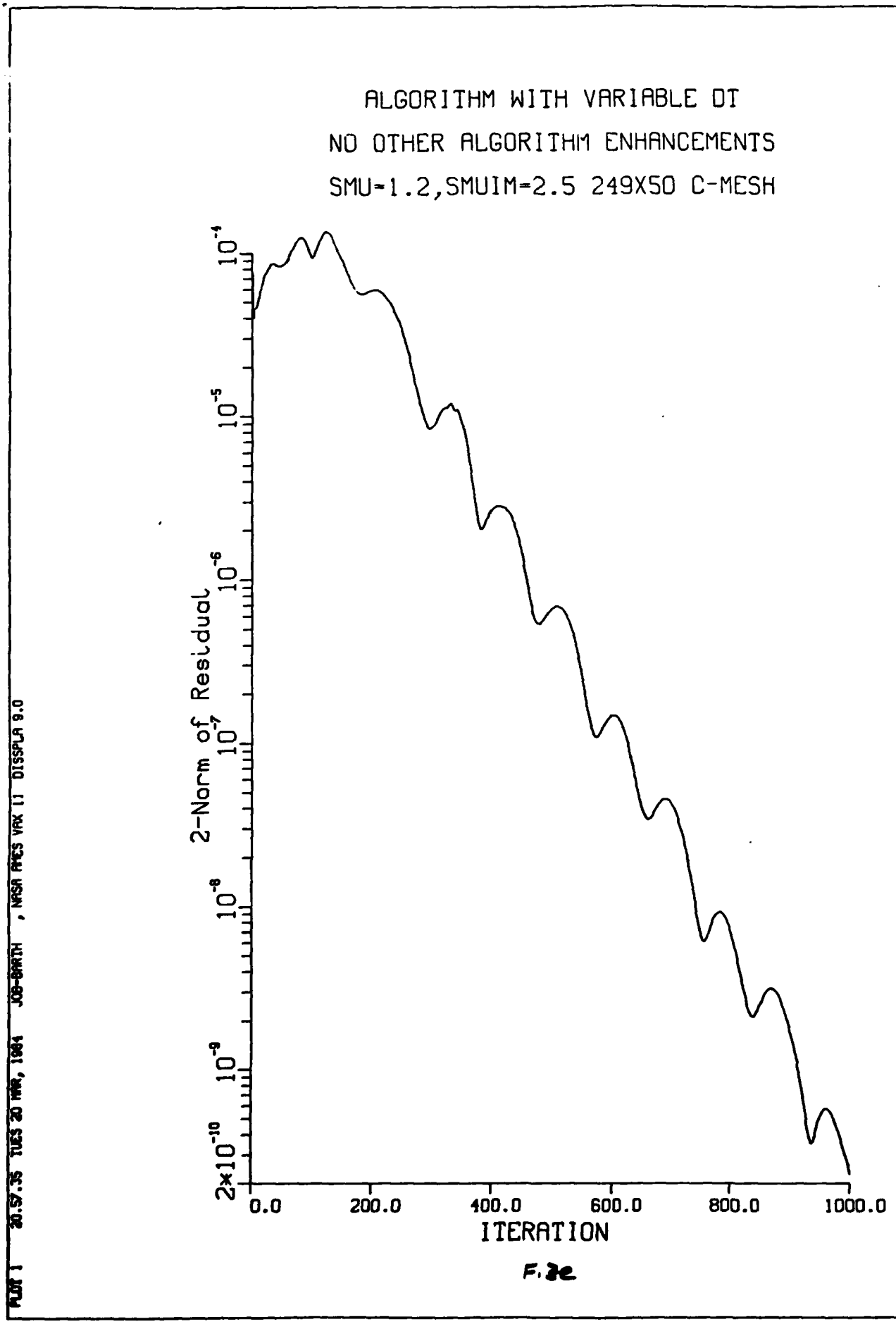
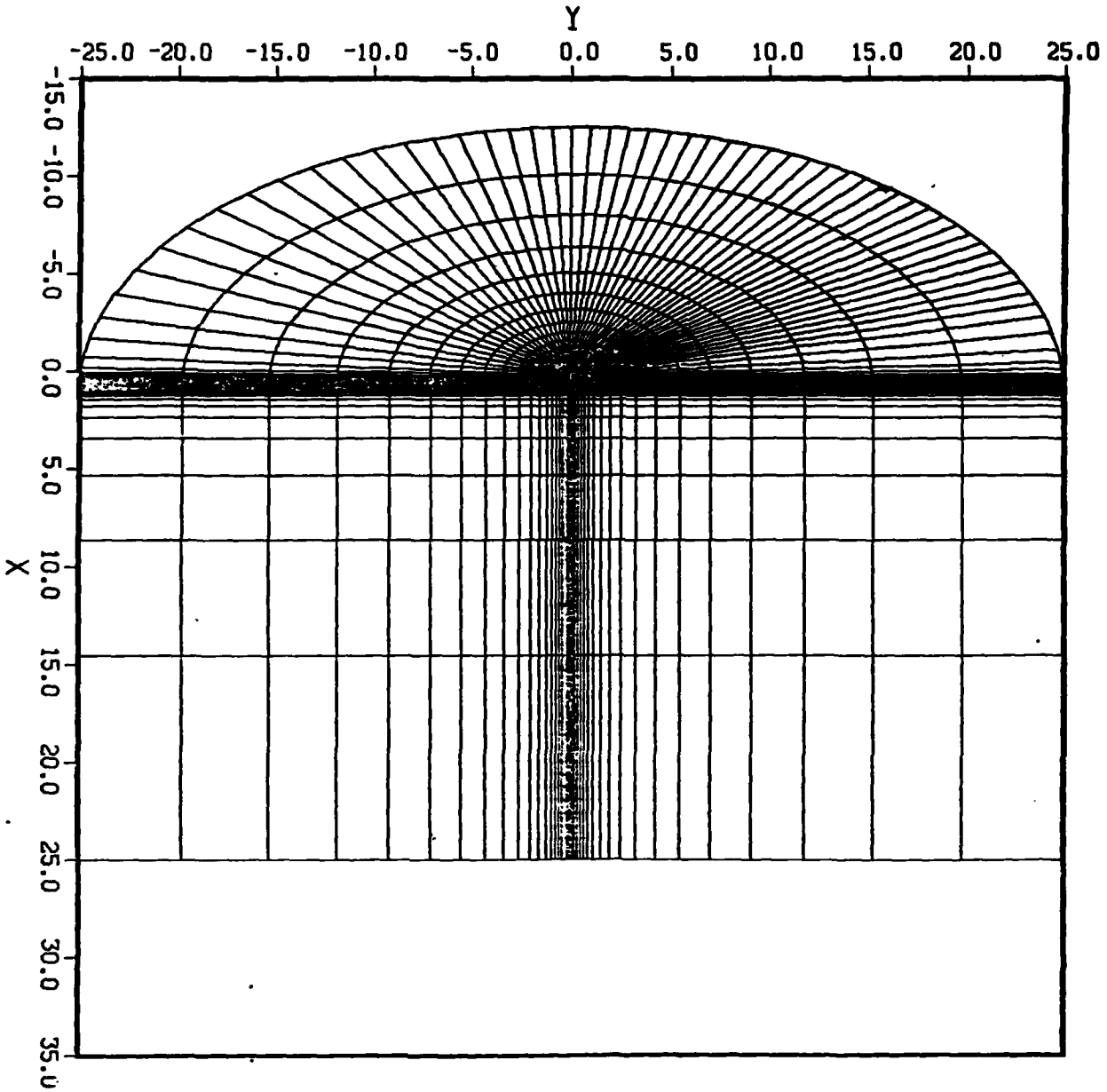
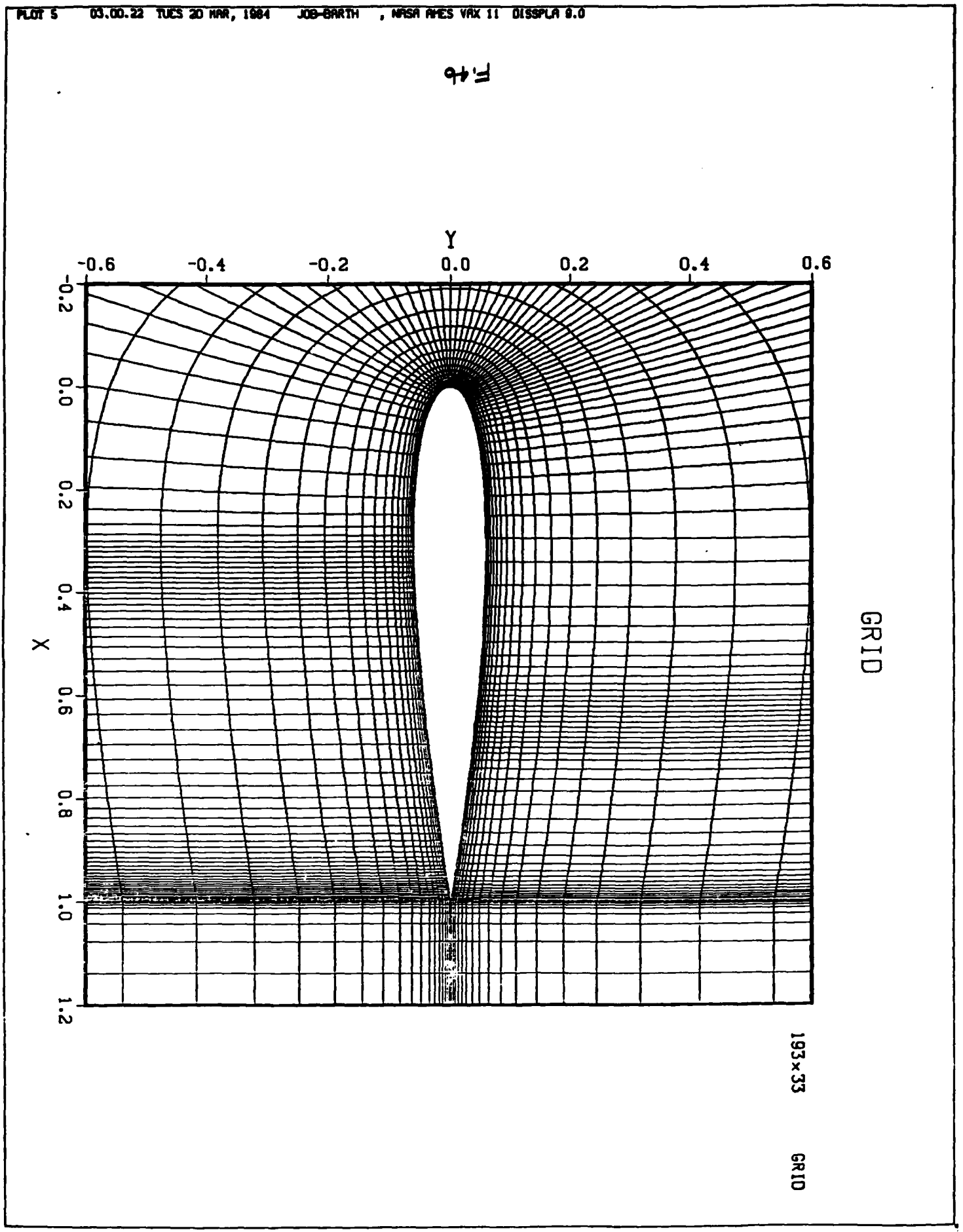

ALGORITHM WITH VARIABLE DT
NO OTHER ALGORITHM ENHANCEMENTS
SMU=1.2,SMUIM=2.5 249X50 C-MESH
2-Norm of Residual
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
2×10^{-10}
0.0
200.0
400.0
600.0
800.0
1000.0
ITERATION
PLOT 1 20:57:35 TUES 20 MAR, 1984 JOB-BARTH , NASA AMES VAX 11 DISSPLA 9.0

Fig 2e

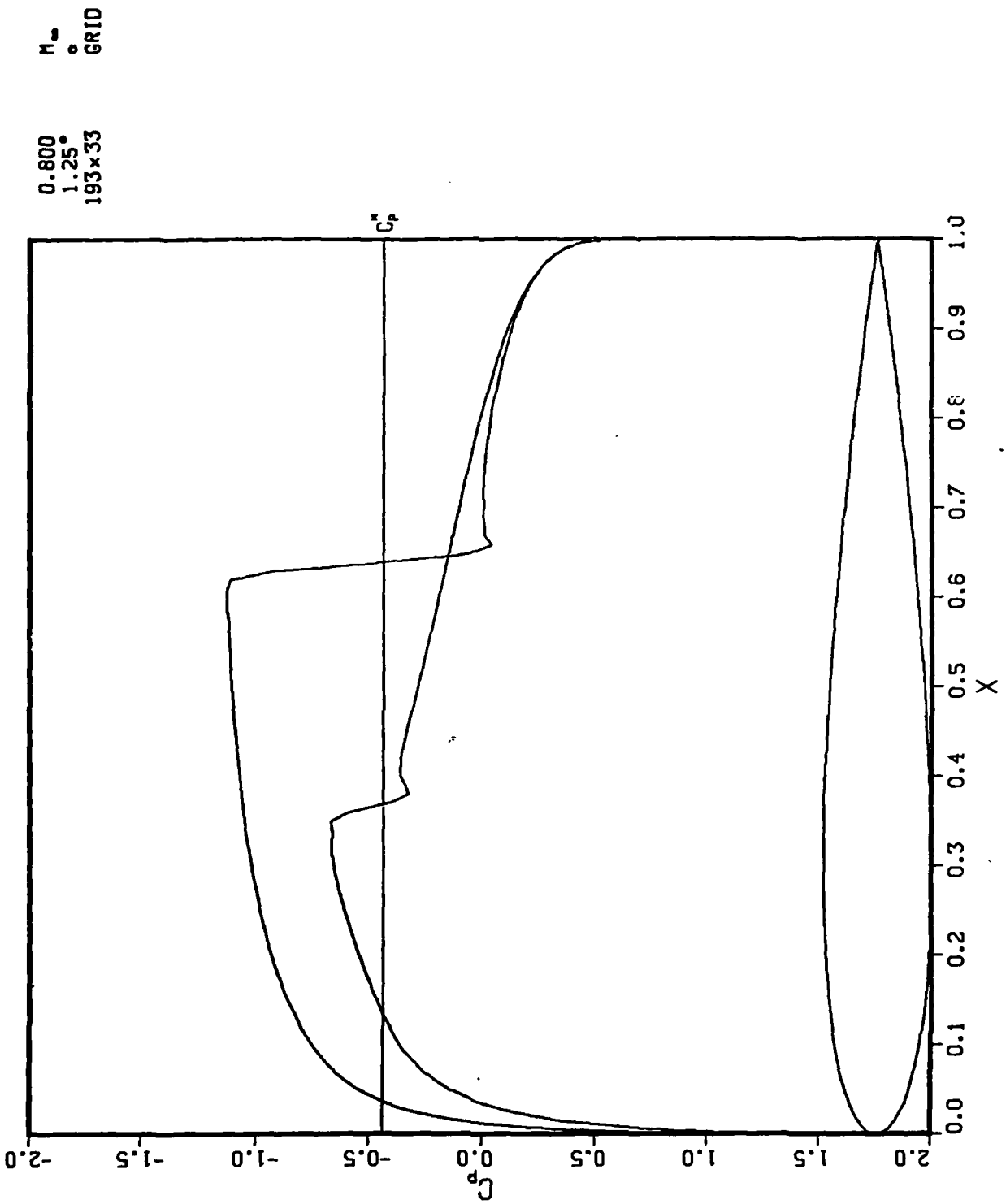


GRID

193×33 GRID


PLOT 5 03.00.22 TUES 20 MAR, 1984 JOB-BARTH , NASA AMES VAX 11 DISSPLA 9.0
Y
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
X
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
GRID
193×33
GRID

PRESSURE COEFFICIENT
ENHANCED ALGORITHM



F.4c

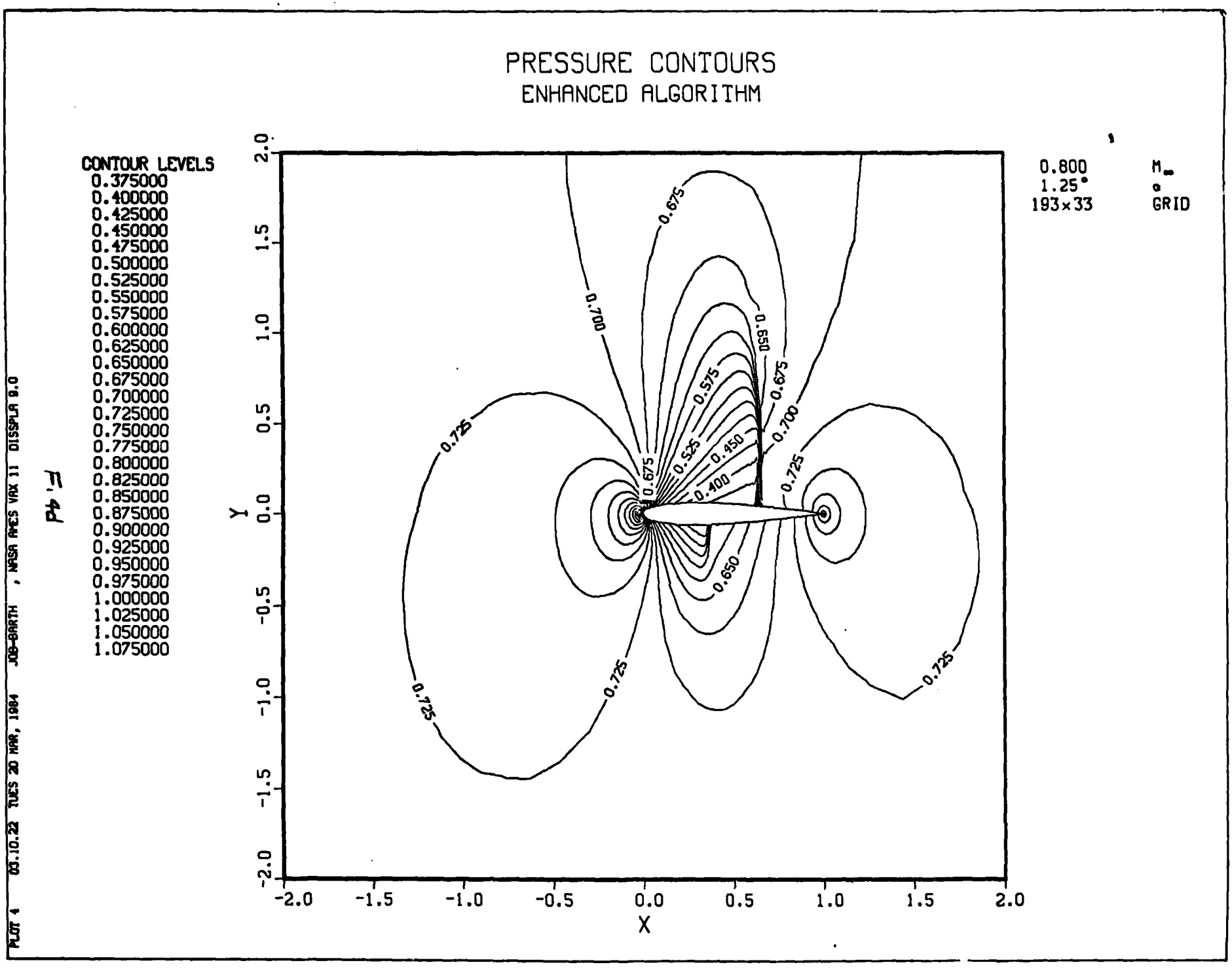
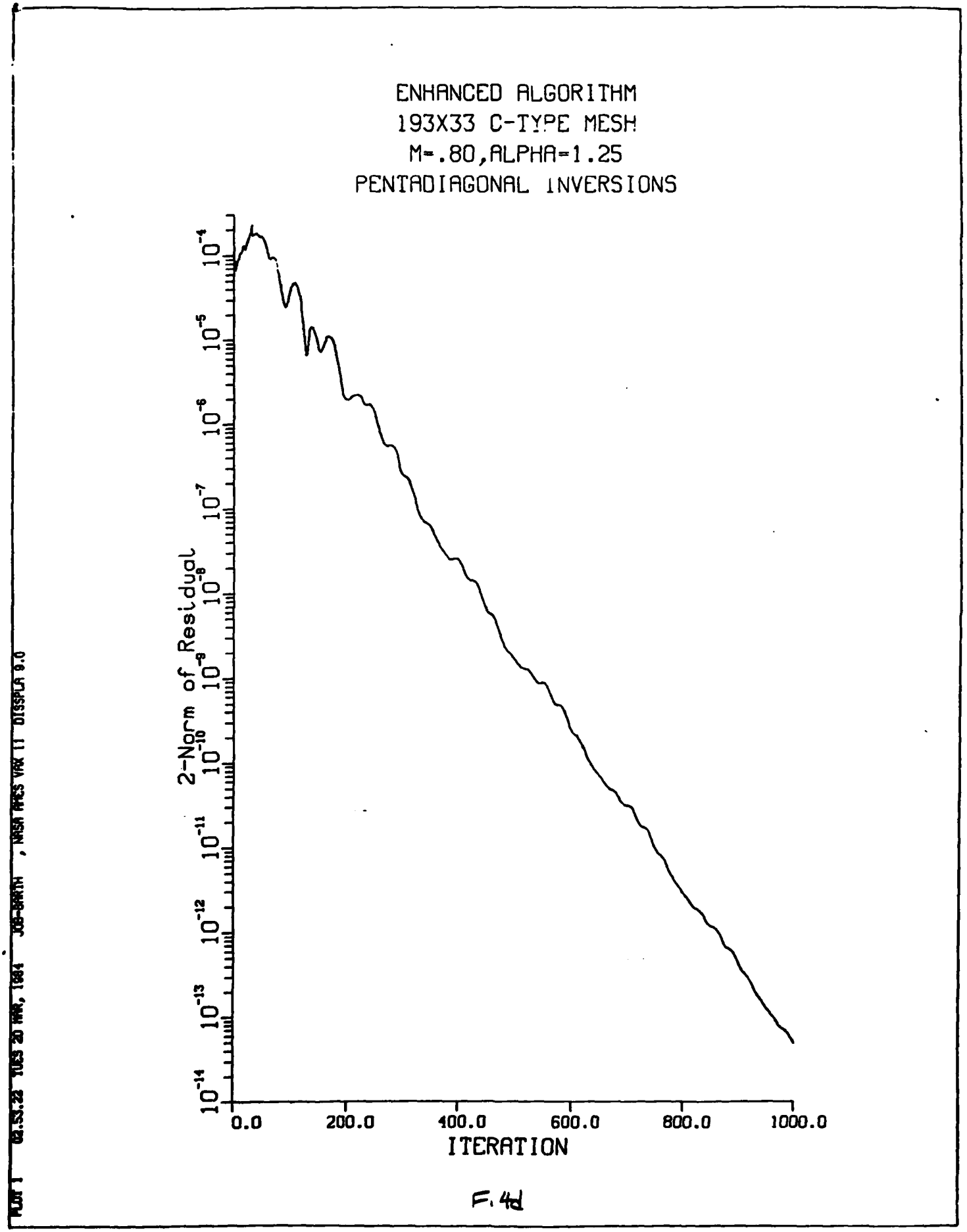

PRESSURE CONTOURS
ENHANCED ALGORITHM
CONTOUR LEVELS
0.375000
0.400000
0.425000
0.450000
0.475000
0.500000
0.525000
0.550000
0.575000
0.600000
0.625000
0.650000
0.675000
0.700000
0.725000
0.750000
0.775000
0.800000
0.825000
0.850000
0.875000
0.900000
0.925000
0.950000
0.975000
1.000000
1.025000
1.050000
1.075000
0.800 M∞
1.25° α
193×33 GRID
Y
X
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
0.675
0.700
0.650
0.575
0.525
0.450
0.400
0.725
0.650
PLOT 4 03.10.22 TUES 20 MAR, 1984 JOB-BARTH , NASA AMES VAX 11 DISSPLA 9.0

Fig 4d


ENHANCED ALGORITHM
193X33 C-TYPE MESH
M=.80,ALPHA=1.25
PENTADIAGONAL INVERSIONS
2-Norm of Residual
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-13}
10^{-14}
0.0
200.0
400.0
600.0
800.0
1000.0
ITERATION

F.4d

END

FILMED

1-85

DTIC